НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему Розробка програмного агенту моніторингу та управління енергетичними потоками будівлі

Виконала: студенка __4__курсу, групи_ТІ-51_

Пироговська Тетяна Володимирівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н. Ковальчук А. М. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент доцент,к.т.н. Баранюк О. В. _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту
Пироговська Тетяна Володимирівна
(прізвище, ім’я, по батькові)

1. Тема роботи Розробка програмного агенту моніторингу та управління енергетичними потоками будівлі _____
_____

керівник роботи_____ доцент, к.т.н. Ковальчук Артем Михайлович_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від 22.05.2019р. № 1325-с

2. Строк подання студентом роботи _____11 червня 2019_____

3. Вихідні дані до роботи роботи: персональний комп’ютер під керуванням операційної системи Microsoft Windows, мови програмування C, C++, C#.

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)_ _ проаналізувати існуючі програмні рішення та можливі засоби реалізації взаємодії, обґрунтувати обрані програмні застосунки та шляхи розробки програмних додатків, розробити програмне забезпечення, розробити користувацький інтерфейс, зробити висновки за результатами роботи_____

5. Перелік ілюстративного матеріалу
_1. Моделювання мультиагентної системи 2. Апаратна архітектура комплексу.

3.Програмна архітектура комплексу. 4.Підключення апаратних частин проекту.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”_14_”__жовтня___2018р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	12.02.19	
2	Розробка архітектури та загальної структури системи	23.04.19	
3.	Розробка структур окремих підсистем	30.04.19	
4.	Програмна реалізація системи	02.05.19	
5.	Оформлення пояснювальної записки	10.05.19	
6.	Захист програмного продукту	18.05.19	
7.	Передзахист	01.06.19	
8.	Захист	19.06.19	

Студент _____ _____Пироговська Т. В._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _____Ковальчук А. М._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Мета роботи — дослідження мультиагентних систем та їх застосування для моніторингу та управління енергетичними потоками будівлію Для створення інтерфейсів використано засоби візуального програмування Windows Forms. Для створення прошивки контролеру було використано мову С, а для створення серверного додатку – С++.

Розроблений апаратно-програмний комплекс забезпечує керування агенту споживача електроенергії.

Записка містить 77 сторінок, 21 рисунків, 2 таблиці, 3 додатки і 13 посилань.

Ключові слова: мультиагентні системи, інтелектуальний агент, система моніторингу та управління, віддалене керування, розподілені системи.

ABSTRACT

This thesis is devoted to the development of intellectual agent for monitoring and controlling of energy flows. For interface development was used Windows Forms. Microcontroller firmware was written in C and server part was written in C ++.

Production of hardware and software complex provides control and monitoring of energy consumption.

Note note 77 pages, 21 figures, 2 tables, 3 documents and 13 poseyn.

Key words: .multiagent systems, intellectual agent, monitoring and control system, remote control, distributed systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

TCP/IP – Transmission Control Protocol/Internet Protocol

UART – Universal Asynchronous Receiver/Transmitter

HAL – Hardware Annotation Library

ACL – Agent Communication Language

KQML – Knowledge Query and Manipulation Language

MAS – Multi-agent systems

DMA – Direct Memory Access

CAN – Controller Area Network

RTC – Real-Time Clock

PWM – Pulse-Width Modulation

ВСТУП

Нові відкриття завжди стимулювались потребою оптимізації робочого процесу для підвищення ефективності роботи. Проте для оптимізації того чи іншого процесу, потрібно постійно обробляти данні системи, в якій він відбувається, та швидко реагувати на зміни у її роботі. Так, для ефективного використання електроенергії в будівлі важливо постійно стежити за роботою системи. Вона потребує даних про поточний стан всіх частин системи для ефективного реагування на виникаючи відхилення у роботі системи.

Одним із рішень цієї проблеми є «розумні будинки». Такі системи є значно ефективнішими, оскільки містять можливість повного контролю системи без присутності оператора, оскільки пропонують роботу за заданими заздалегідь сценаріями роботи. Проте такий спосіб виключає можливість реагувати на поточні зміни системи, що може призвести до збоїв роботи системи. Це рішення не дає точного моніторингу стану системи та не локалізує причину несправності або збою роботи. Вона не здатна корегувати роботу відповідно до зміни стану системи.

Іншим рішенням даної проблеми є мультиагентні системи. Мультиагентні системи – системи, що складаються з інтелектуальних агентів, кожен з яких має свою визначену функцію у системі. Основними характеристиками такої системи є автономність кожного агента (його незалежність), обмеженість уявлення (агент не має уявлення про всю систему та зосереджений на власній ланці) та децентралізація системи (немає агентів чи програм, які керують усією системою). Такий підхід до створення системи дозволяє ефективний моніторинг кожної ланки та її незалежну роботу, що створює безпечну систему, у якій відключення або погана робота однієї з ланок не припиняє роботу усієї системи.

Таку технологію можна ефективно застосувати разом з енергетичними системами Mictogrid. Microgrid-системи – енергетичні системи, які містять розподілені генератори, ресурси відновлювальної енергії, енергозберігаючі кластери, які координовано працюють для забезпечення надійного постачання енергії та раціонального її використання. Вони можуть бути як частиною більшої енергетичної

системи, так і працювати автономно. Тобто Microgrid-системи та мультиагентні системи мають схожу ідею, закладену у своїй архітектурі, що дозволяє використовувати їх сумісно.

Тому для рішення проблеми було запропоновано створити апаратно-програмний комплекс, як частину мультиагентної системи, яка керуватиме роботою Microgrid-системи енергопостачання. Такий агент має вміти опрацьовувати дані своєї ланки та керувати її окремими частинами.

Було поставлено завдання: проаналізувати мультиагентні системи, розробити та скласти апаратну частину системи, розробити спосіб аналізу та передачі на персональний комп’ютер даних, розробити графічний додаток, який відображатиме роботу агенту.

Для реалізації даного програмного продукту було обрано платформу Windows Form, мови програмування C, C++, C#.

1 АНАЛІЗ МУЛЬТИАГЕНТНОЇ СИСТЕМИ МОНІТОРИНГУ ТА УПРАВЛІННЯ ЕНЕРГЕТИЧНИМИ ПОТОКАМИ БУДІВЛІ

У світі сучасної енергетики все популярнішими стають розподілені системи, оскільки вони забезпечують надійне постачання електроенергії. Існує багато рішень поставленої задачі енергоефективної та само відновлюваної розподіленої системи і одним з найцікавіших є ідея Microgrid-системи.

Microgrid-системи можуть бути перспективною енергетичною системою, яка стосується технологій відновлюваної енергії, що супроводжують необхідне посилення розповсюдження розподілених енергетичних ресурсів, особливо дрібномасштабних теплових та електричних і дрібномасштабних відновлюваних джерел енергії. Такі системи зазвичай включають: мікротурбіни, паливні елементи, фотоелектричні елементи, сонячні теплові масиви та установки вітрових турбін, а також, регулювання накопичення, регулювання навантаження, регулювання потужності та напруги та рекуперації тепла повинні бути згруповані в Microgrid-систему. Мікросетки можуть задовольняти витрати, ефективність і екологічні переваги; та вимоги до безпеки, якості, надійності та доступності, що надаються на виробництві на місці, досягається шляхом включення сучасних засобів управління та функціонування з певною автономією [1].

Для ще ефективнішого використання таких систем, в їх управлінні можна використовувати принцип мультиагентних систем [2].

1.1 Опис мультиагентної системи

Мультиагентна система – комп’ютеризована система, яка складається з незалежних інтелектуальних агентів. Головною ідеєю є розділення складної системи на окремі ланки, кожна з яких контролюватиметься своїм агентом. Ці агенти мають не повне уявлення про систему або зовсім не знають про будову системи, у якій працюють.

Метою мультиагентних систем є розуміння того, яким чином можна координувати незалежні процеси. Мультиагентна система складається з набору комп'ютерних процесів, які відбуваються одночасно, тобто декількох агентів, які існують одночасно, спільно використовують спільні ресурси і спілкуються один з одним. Ключовим питанням у мультиагентних системах є формалізація координації між агентами.

Дослідження агентів включають у себе такі проблематики:

— Прийняття рішень: які механізми прийняття рішень доступні агенту? Що таке зв'язок між їхніми сприйняттями, уявленнями і діями?

— Контроль: які ієрархічні відносини існують між агентами? Як вони синхронізуються?

— Зв'язок: які повідомлення вони посилають один одному? Який синтаксис виконують ці повідомлення?

Мультиагентні системи можуть бути застосовані до штучного інтелекту. Вони спрощують вирішення проблем, розділяючи необхідні знання на підрозділи, до яких асоціюється незалежний розумний агент, і координуючи діяльність агентів. Таким чином, ми посилаємося на розподілений штучний інтелект.

Мультиагентні системи можуть виявляти самоорганізацію, а також парадигми самоврядування та інші контрольні парадигми та пов'язані з ними складні поведінки навіть тоді, коли індивідуальні стратегії всіх їхніх агентів прості. Агенти можуть обмінюватися знаннями з використанням будь-якої узгодженої мови, в межах обмежень протоколу зв'язку системи, підхід може призвести до загального поліпшення. Мови прикладу - мова маніпулювання запитами знань (KQML) або мова комунікації агента (ACL) [3].

Така система має значні переваги над іншими, оскільки забезпечує безперервну роботу навіть у випадку неполадок у роботі одного з агентів системи. Інший спосіб

будування передбачає наявність ієрархічної будови, де функціональність усієї системи залежна від головного агента, який обробляє дані усіх ланок та або приймає рішення на основі цих даних, або контролюється оператором.

Приклад такої системи можемо побачити на рисунку 1.1. Через свою ієрархічну будову система дуже чутлива до поломок головного агенту. У випадку будь-яких фатальних помилок, непередбачених програмістом, уся система припиняє роботу, тому така система потребує постійної присутності оператора, який міг би контролювати та підтримувати роботу системи.

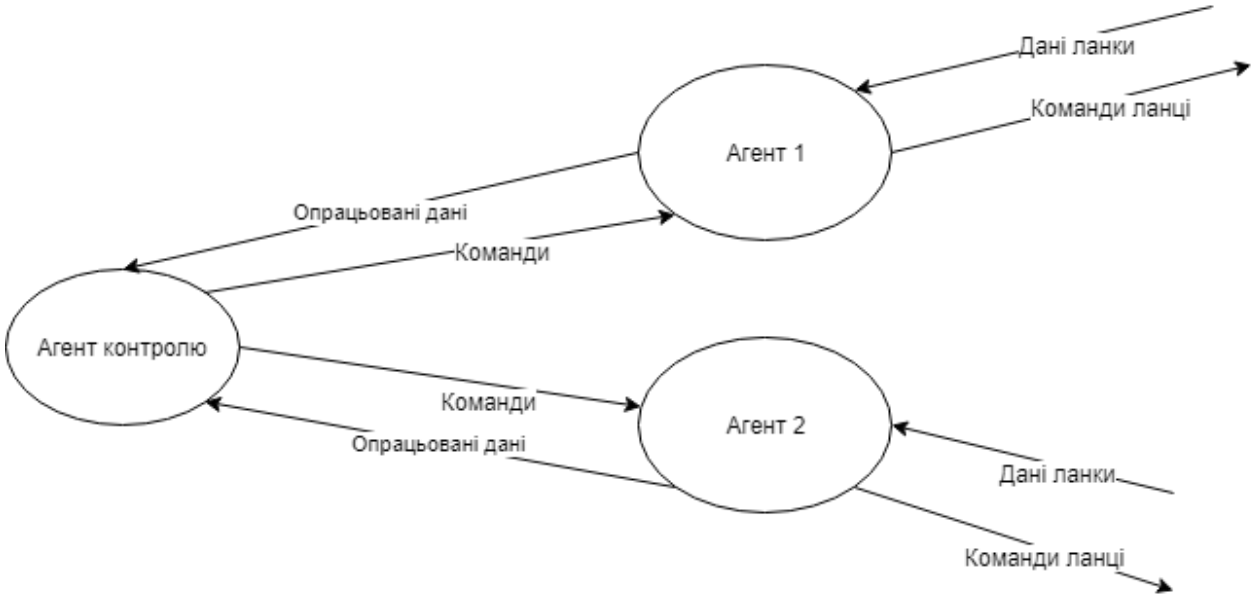


Рисунок 1.1 – Класичне представлення системи управління та моніторингу.

На відміну від систем ієрархічної будови, мультиагентні системи значно надійніші та потребують значно менше робочої сили. Цей метод може бути використаний для моніторингу промислового процесу, наприклад, коли приймається загальне для системи рішення – координація роботи декількох спеціалізованих інтелектуальних агентів, а не єдиного «всезнаючого» елементу.

Проводяться фундаментальні дослідження проблем, пов’язаних з поданням агентських рішень і протоколів для комунікації. Основними додатками для MAS є телекомунікації, Інтернет та фізичні агенти, такі як роботи.

Дослідження мультиагентних системи досягло прогресу в AI, апаратних засобах і сенсорних технологіях, що призвело до того, що технології агентів успішно застосовуються до реальних промислових проблем. Проект VOLTTRON був підтриманий американським департаментом енергетики для передачі програмної платформи VOLTTRONTM до Tranformative Wave. Крім того, департамент забезпечує Tranformative Wave технічною підтримкою для розробки продуктів і послуг, що підвищують ефективність роботи будівель і стійкість електричних мереж. Цей проект вказує на те, що промисловість приймає агентські технології [4].

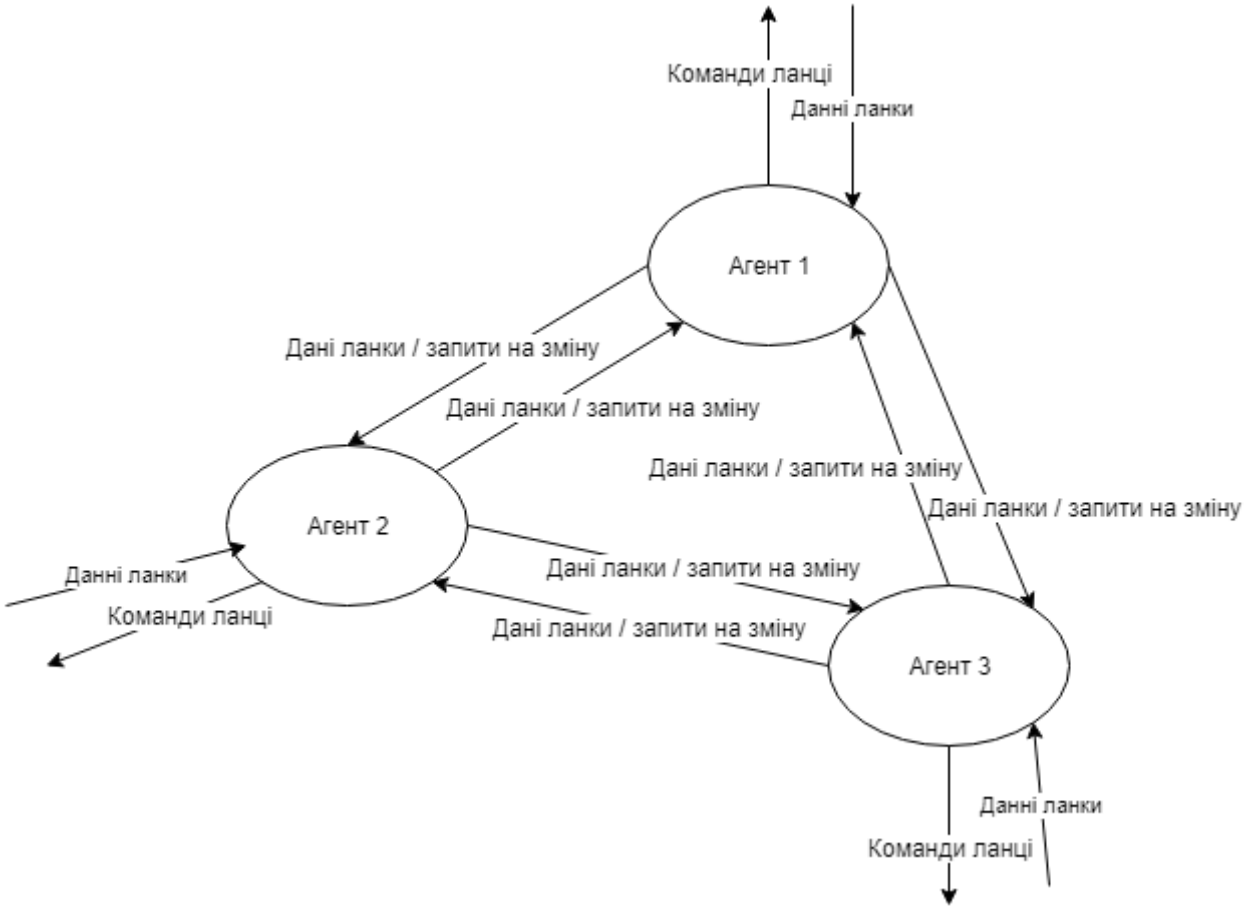


Рисунок 1.2 – Приклад мультиагентної системи

На рисунку 1.2 зображено приклад простої мультиагентної системи, яка складається з трьох незалежних агентів. Кожен з них транслює данні про свою ланку, або запити на зміну роботи ланки іншого агенту. Проте вже сам агент обирає чи задовольняти запит від іншого агенту, зважуючи на поточний стан своєї ланки.

Дослідження мультиагентної системи в основному орієнтовані на архітектуру системи, консенсусний алгоритм, розподілену оптимізацію та програмні засоби для моделювання та впровадження. Передбачається, що агентські технології продовжуватимуть розвиватися в майбутньому. Крім того, більша кількість галузей промисловості шукають способи використання таких систем, вказуючи на те, що агентські технології досягають значних успіхів у комерційному використанні.

Процес розробки програмного забезпечення мультиагентних систем вимагає технічної підтримки з обраної платформи. Важливо зрозуміла та детальна документація та зручний та інтуїтивний інтерфейс для користувача. Ефективність розробки може бути значно підвищена, якщо надається хороший комплект розробки програмного забезпечення (SDK). Після розгортання мультиагентної системи, обслуговування стає довгостроковим питанням, що стосується складного середовища та географічно широко розповсюджених вузлів. Спеціалізовані та професійні послуги з підтримки розподілених апаратних пристроїв та програмних компонентів є критичними в майбутньому.

1.2 Опис агенту мультиагентної системи

Для проектування агенту мультиагентної системи потрібно дотримуватись головних критеріїв системи:

— Автономність: агент має бути або частково, або повністю незалежним.
— Обмеженість уявлення: агент не має повного представлення про будову системи.
— Децентралізація: у системі відсутній головуючий агент, який керує іншими; агенти рівні між собою.

Тобто агент мультиагентної системи бачить лише свою ланку та її стан. У разі потреби зміни роботи якоїсь іншої ланки, агент має відправити запит до агентів на цю

зміну. Агент, який відповідає за ту ділянку, відреагує на запит, і залежно від поточного стану, або задовольнить запит, або відкине його.

Наприклад, ланка управління вентиляційної системи (Агент 1) потребує подання струму від ланки з генератором струму (Агент 2). Агент 1 надсилає запит на подання струму на свою ланку, та передає його усім агентам у мережі. Агент 2, як і всі інші агенти, отримує відправлений запит, та обробляє його. Перевіряючи поточний стан, агент 2 виявляє, що струм на ланку не можливо подати через загальну відсутність струму у всій ланці. Агент 2 відхиляє запит та інформує агент комунікацій про неполадку у роботі генератора.

Тобто головними вимогами до агенту є:

— Можливість зв’язку через протокол TCP/IP для відправки та отримання даних від інших клієнтів, які знаходяться в одній мережі.

— Зчитування даних системи (зчитування значень датчиків та пристроїв системи).

— Опрацювання отриманих даних та їх збереження.

— Побудова аналітичної звітності роботи системи.

2 АРХІТЕКТУРА АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

Концепції мультиагентної системи та Microgrid-системи можна застосувати для управління та моніторингу до різноманітних систем різного призначення. Зокрема, такий спосіб побудови системи можна успішно застосувати для ефективного використання електроенергії.

Тому для створення програмного продукту потрібно проаналізувати інженерні системи будівлі та визначити роль агенту у загальній мультиагентній системі, для формування вимог, а звідси, і архітектури агенту, як апаратної, так і програмної.

2.1 Архітектура мультиагентної системи

На рисунку 2.1 зображено сукупність інженерних систем будівлі, згрупованих у мультиагентну систему. Керування над ланками мережі та над процесами, було розділені на незалежні частини, кожна з яких може функціонувати незалежно, або майже незалежно.

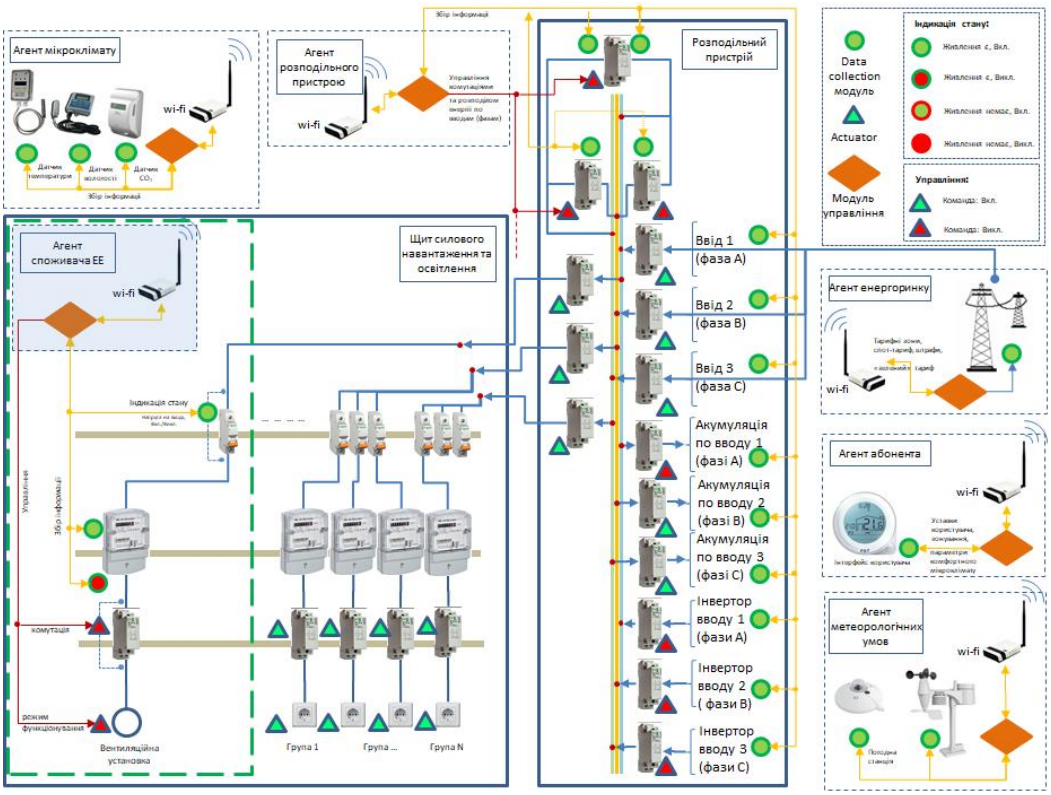


Рисунок 2.1 – Мультиагентна система інженерних систем будівлі

Робота присвячена розробці Агенту споживача електроенергії. Область, яка підпадає під контроль даного агенту, виділена зеленою рамочкою. Отже ланка агенту під'єднана до загальної мережі електропостачання та регулює свою роботу відповідно до роботи агент розподільного пристрою. Цей пристрій контролює подання електроенергії у мережі, та приймає рішення про подання або приривання подання енергії на усі підконтрольні йому ланки.

Не менш важливим є агент енергоринку, який надсилає рекомендації з використання електроенергії, відповідно до аналізу вартості електроенергії у різні години доби.

У системі також присутні інші різноманітні агенти, кожен з яких виконує свою функцію. Наприклад, агент мікроклімату аналізує стан приміщення та відправляє відповідні запити на зміну до відповідних агентів: запит на зменшення температури чи запит на зменшення вологості агенту вентиляції приміщення. Агент, отримавши запит, аналізує можливість включення установки. В залежності від критичної температури та від рекомендації агенту моніторингу енергоринку, агент вентиляційної системи приймає рішення про роботу ланки.

Ланка агенту споживача складається з декількох елементів: рубильник, лічильник, реле та саму вентиляційну установку.

Для моніторингу роботи такої системи потребується 4 датчики струму. Їх потрібно розмістити в наступних точках, для ефективного моніторингу стану мережі: два датчики потрібно розмістити до і після рубильника та два до і після лічильника. Такий спосіб розміщення дозволить визначати на якій саме ланці є проблеми з постачанням струму та дозволить швидко вирішувати проблему енергопостачання.

Також агент має мати комутаційне підключення до самої установки, для забезпечення контролю над нею.

Для зв'язку з іншими агентами мережі, агент має мати з'єднання Інтернет для створення TCP/IP зв'язку.

2.2 Архітектура апаратної частини системи

Проаналізувавши вимоги системи до агенту, було вирішено створити наглядний стенд, який демонструватиме роботу мультиагентної системи.

Апаратна частина складається з двох елементів – сервера та контролера. Контролер збирає дані з датчиків та інших пристроїв, а також керує елементами системи. Сервер в свою чергу збирає дані з контролера, аналізує їх та відповідно до них передає команди контролеру.

Аналізовані данні передаються клієнтському додатку для візуального відображення. На рисунку 2.2 зображена схема апаратного забезпечення агенту.

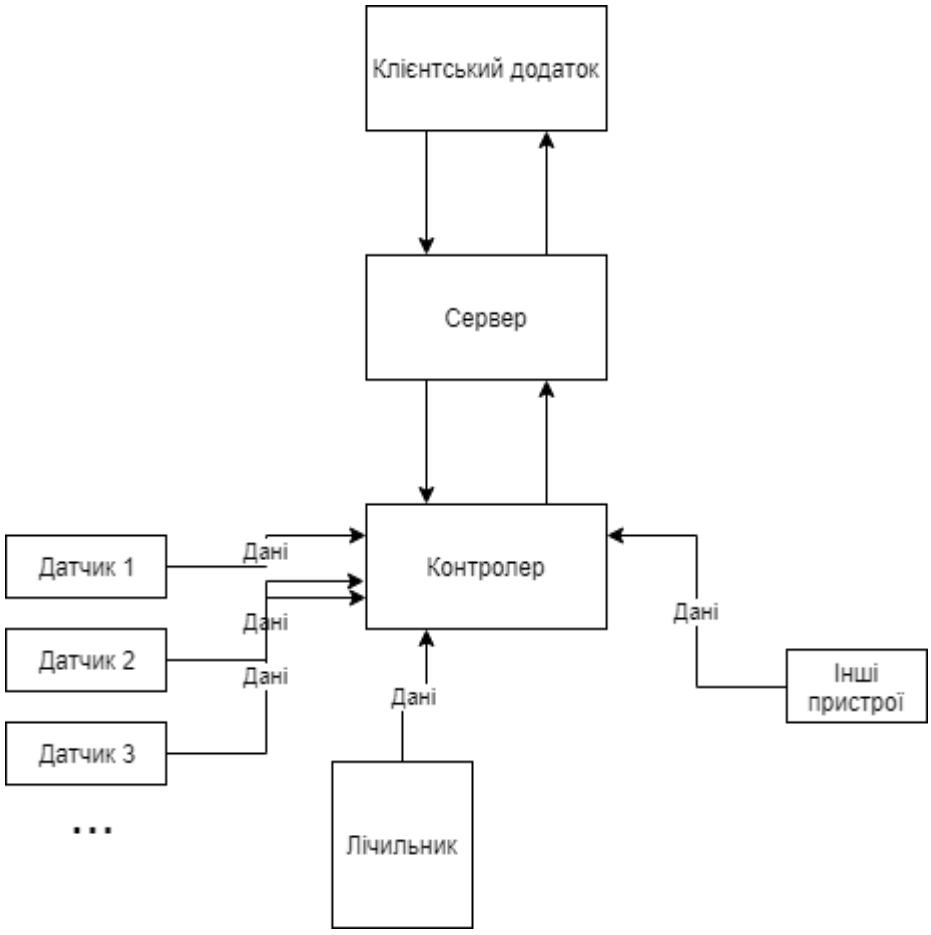


Рисунок 2.2 – Архітектура апаратної частини комплексу

Сервером слугуватиме одноплатний комп’ютер. Він дозволяє не тільки налаштувати передачу даних через TCP/IP-протокол, а також має можливість

підключення девайсів по UART-протоколу. Це дозволяє нам підключити контролер та настроїти обмін даними серверу та клієнт-додатку.

Клієнтський додаток – комп'ютерний додаток, який має запускатись на операційній системі Windows. Даний додаток є лише інтерфейсом, якій візуалізує дані, отримані з серверу.

2.3 Архітектура програмної частини системи

Програмна частина системи складається з трьох частин: контролеру, серверу та клієнт-додатку (рисунок 2.3).

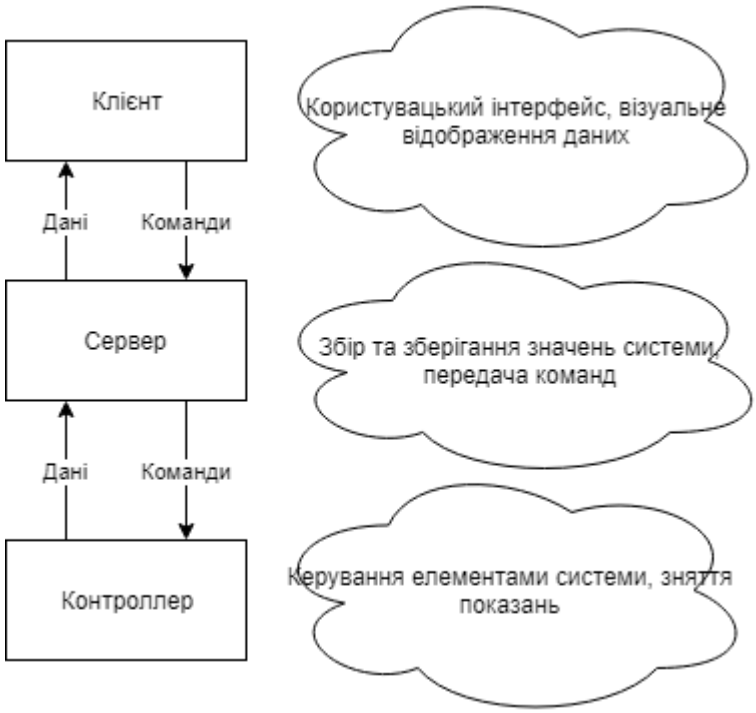


Рисунок 2.3 – Архітектура програмної частини комплексу

Контролер має кожну секунду знімати показники з приєднаних до нього пристроїв, а також реагувати на команди, сформовані сервером.

Сервер є головною частиною агенту, тому важливо забезпечити його постійну роботу. Тобто, у випадку від'єднання іншого клієнту від розроблюваного, він має продовжувати працювати без непередбачуваної зупинки програми.

Тому у проектуванні серверу, було вирішено розробити його так, щоб кожен агент є сервером і клієнтом одночасно. Сервер відкриває порт прослуховування інших клієнтів, та чекає з’єднання для отримання даних. Коли агенту потрібно відправити дані, він створює тимчасового TCP-клієнта, який підключається, передає дані та видаляється. Такий спосіб зв’язку забезпечує стабільність та роботу незалежно від того, чи підключений агент до іншого чи ні. Також це забезпечує можливість одночасного обміну даних з різними агентами.

Клієнт-додаток використовується лише як візуалізація отриманих даних від агенту, та спосіб комунікації з ним.

3 АНАЛІЗ ТА ОБҐРУНТУВАННЯ РЕАЛІЗАЦІЇ АПАРАТНО-ПРОГРАМНОГО КОМПЛЕКСУ

У ході розробки програмного продукту було використана модель бізнес логіки програмного забезпечення. Цей підхід дозволяє розширити функціонал мультисистеми у ході подальшого циклу життя програмного продукту без затрат на перепроектування. Також такий підхід дозволяє паралельну роботу декількох програмістів.

Після постановки задачі, було сформульовано основні критерії, які має задовільнити програмна частина системи:

- Архітектура розробленого програмного продукту має забезпечувати можливість швидкого нарощування функціоналу.
- Розширення наявних можливостей системи без кардинальних змін інтерфейсу і роботи користувача з системою.

Аналіз вимог до програмного забезпечення показав, що найкращим вибором побудови архітектури, який задовольняв би перелічені вимоги є принцип побудови х використанням об’єктно-орієнтованого програмування, за допомогою якого, можна досить легко вирішити проблеми з майбутнім розширенням системи. Такий спосіб організації системи має багато переваг:

- Швидкий пошук помилок і їх виправлення: на відміну від програм монолітного комплексу, елементи системи легко редагувати, та їх зміна не впливає на оточуючі елементи.
- Інтерфейс модуля: дозволяє надавати певний набір методів для використання можливостей у інших модулях, що робить можливою стандартизацію протоколів міжмодульної взаємодії.
- Компіляція: було вибрано інтерпретовану мову розробки, що робить процес компіляції та тестування значно швидшим.

Реалізація такої системи потребує попереднього аналізу та проектування для забезпечення правильної взаємодії між модулями та чіткого розподілу завдань між модулями. Результат виконаної роботи має забезпечувати функціональність на рані бібліотеки класів, для можливості подальшого використання, тобто модульний підхід є ідеальним способом проектування системи.

Також при розробці програмного продукту були застосовані принципи об’єктно-орієнтованого програмування.

Об’єктно-орієнтоване програмування (ООП) є моделлю мови програмування, в якій програми організовані навколо даних або об’єктів, а не функцій і логіки. Об’єкт може бути визначений як поле даних, яке має унікальні атрибути і поведінку. Такий підхід є повною протилежністю історичного підходу до програмування, де акцент робився на написанні логіки, а не на тому, як визначати дані логіки.

Перший кроком в ООП полягає в тому, щоб ідентифікувати всі об’єкти, якими програміст хоче маніпулювати, і як вони пов’язані один з одним. Такий процес більш відомий як моделювання даних. Як тільки об’єкт відомий, він узагальнюється як клас об’єктів, який визначає тип даних, які він містить, і будь-які логічні послідовності, які можуть маніпулювати нею. Кожна окрема логічна послідовність відома як метод, і об’єкти можуть спілкуватися з чітко визначеними інтерфейсами, які називаються повідомленнями.

Тобто, ООП зосереджується на об’єктах, якими розробники хочуть маніпулювати, а не на логіці, необхідній для маніпулювання ними. Такий підхід до програмування добре підходить для великих, складних і активно оновлюваних або підтримуваних програм. Завдяки організації об’єктно-орієнтованої програми, цей метод також сприяє спільному розвитку, де проекти можна розділити на групи. Додаткові переваги ООП включають повторне використання коду, масштабованість і ефективність.

Об’єктно-орієнтоване програмування базується на наступних принципах [6]:

— Інкапсуляція: Реалізація і стан кожного об’єкта перебувають у приватній власності в межах певної межі або класу. Інші об’єкти не

мають доступу до цього класу або повноважень для внесення змін, але можуть викликати лише список загальнодоступних функцій або методів. Ця характеристика приховування даних забезпечує більшу безпеку програми і дозволяє уникнути ненавмисного пошкодження даних.

— Абстракція: Об’єкти лише розкривають внутрішні механізми, які мають відношення до використання інших об’єктів, приховуючи будь-який непотрібний код реалізації. Ця концепція допомагає розробникам робити зміни та доповнення з часом.
— Успадкування: Можуть бути призначені відносини і підкласи між об’єктами, що дозволяє розробникам повторно використовувати загальну логіку, зберігаючи при цьому унікальну ієрархію. Це властивість ООП змушує більш ретельний аналіз даних, скорочує час розробки і забезпечує більш високий рівень точності.
— Поліморфізм: Об’єктам дозволено брати більше однієї форми залежно від контексту. Програма визначить, який зміст або використання необхідно для кожного виконання цього об’єкта, зменшуючи необхідність дублювання коду.

Завдяки цим принципам об’єктно-орієнтоване програмування гнучке та зручне. Програмний продукт, розроблений за цими принципами є відображенням предметної області як ієрархічних об’єктів, взаємодія яких відбувається так само, як і в предметній області. Даний метод надає більше контролю над поведінкою програми за рахунок об’єднання понять та їх реалізації.

Вибір цього методу програмування продиктований необхідністю створення простого та надійного продукту, який легко буде інтегрувати з іншими програмами.

У якості мов програмування, які б задовольняли вимоги об’єктно-орієнтованого програмування було вирішено використовувати C++ (для обробки даних на стороні пристрою) та C#(для імітації стороннього агенту).

3.1 Оцінка вимог

Проаналізувавши архітектуру системи та функції агенту в ній, було сформульовано головні вимоги до створюваного агенту:

— Агент має збирати інформацію стану системи енергопостачання.
— Агент має вміти збирати показання приладів, підключених до системи енергопостачання. Наприклад, знімати данні з лічильнику.
— Агент повинен мати засоби аналізу та обробки отриманих даних.
— Агент повинен мати спосіб зв’язку з іншими агентами системи.

Для демонстрації роботи розробленого агенту, було вирішено створити інтерфейс-додаток, з допомогою якого можна буде імітувати роботу іншого агенту, а тобто, обмін даними між агентами.

3.2 Вибір мікроконтролера

За вимогами до апаратної частини комплексу, агент має знімати дані з присутніх у системі датчиків. Створюваний апаратний продукт повинен вміти працювати з датчиками струму та з лічильником. Також має мати декілька комутаційний виходів, до яких можна буде підключити контроль різними пристроями. Більшість мікроконтролерів мають дані можливості, тому для вибору мікроконтролеру, потрібно уважно проаналізувати представлені продукти та скласти точний список вимог.

На ринку мікроконтролерів великою популярністю користуються 2 марки контролерів – Arduino та STM32. Для порівняння було відібрано дві моделі, які задовольняють апаратні вимоги за помірну ціну – STM32F103C8F6 та Arduino Nano.

Порівняльну характеристику подано у таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика контролерів Arduino та STM32

Характеристики	**STM32F103C8T6**	**Arduino Nano**
Частота контролера	72 МГц	16МГц
Пам’ять програм	64 кБайт	32 кБайт
Живлення	3.3 v	5 v
Оперативна пам’ять	20 кБайт	2 кБайта
USB	присутній	відсутній
DMA	присутній	відсутній
CAN	присутній	відсутній
RTC	присутній	відсутній
PWM	до 16 каналів	до 6 каналів
UART	3 шт.	відсутній
Прошивка через USB	відсутній	присутній
Ціна	85 ₴	128 ₴

Хоч STM32F103C8T6 трохи дорожчий за Arduino та не має можливості прошивки через USB, контролер забезпечує набагато більше можливостей. Контролер забезпечує прямий доступ до пам’яті, має CAN-шину та вбудований годинник. Окрім того, дана модель має 3 UART виходи та велику оперативну пам’ять – 20 кБайтів. Також контролер сімейства STM32 забезпечує велику кількість GPIO виходів, 16 з яких можуть використовуватись у режимі ШІМ. Отже за технічними характеристиками, STM32 має значно більше переваг ніж подібна модель з сімейства Arduino.

Якщо ж аналізувати спосіб програмування та середовища розробки, Arduino має значно більше переваг. Сімейство Arduino має власну екосистему, зручну і зрозумілу навіть початківцям. Також існує безліч бібліотек у вільному доступі, які мають широкий спектр функціоналу, що значно спрощує розробку прошивки мікроконтролеру. STM32 в свою чергу має вищий поріг входження нових бібліотек,

а отже і забезпечує значно меншу кількість готових бібліотек. Але з іншого боку, такий високий поріг входження забезпечує надійність і передбачуваність роботи кожної функції.

На відміну від Arduino, розробка на STM32 виконується на мові С, стандарту С99, що може створювані певні незручності при переході з об’єктно-орієнтованих мов програмування.

Габарити обох контролерів однакові.

На основі аналізу характеристик обох контролерів, було прийнято рішення використовувати STM32F103C8T6, оскільки даний контролер забезпечує ширший спектр функціоналу за приблизно ті самі гроші.

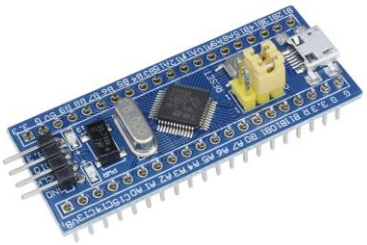


Рисунок 3.1 – Мікроконтролер STM32F103C8T6

Розглянемо будову обраного мікроконтролеру – STM32F103C8T6. Як видно з рисунку 3.2, він забезпечений двома індикаторами – індикатор живлення та вільний світлодіод PC13, який можна запрограмувати.

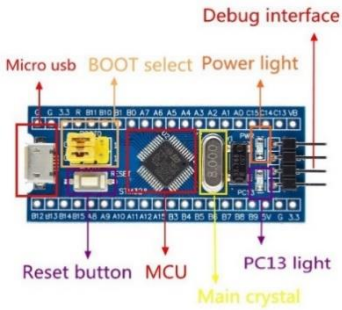


Рисунок 3.2 – Будова мікроконтролеру

Контролер має Micro usb, який забезпечує можливість обміну даних мікроконтролеру та комп’ютеру. Такий спосіб зв’язку зручний для відладки прошивки, для перевірки значень, які транслює мікроконтролер.

Для перепрошивки та відладки, мікроконтролер забезпечений спеціальним виходом.

Boot Select дозволяє вибрати потрібний режим завантаження контролеру. Значення та їх пояснення, наведені у таблиці 3.2.

Таблиця 3.2 – Конфігурації Boot піну.

Піни режиму завантаження		Режим завантаження	Пояснення
BOOT1	BOOT0		
X	1	User Flash memory	Звичайний режим роботи, програма запускається з Flash.
0	1	System memory	Запускається системний завантажувач. Це режим прошивки Flash-пам’яті мікроконтролеру.
1	1	Embedded SRAM	Програма завантажується в оперативну пам’ять контролеру для збереження ресурсу програмування Flash-пам’яті.

Розглянемо GPIO-виходи мікроконтролеру. Для обміну даними з одноплатним комп’ютером, ми потребуємо UART-з’єднання. Як видно з рисунку 3.3, мікроконтролер має 3 таких з’єднання: 1 з лівого боку, 2 з правого. Для управління вентиляційною установкою, ми потребуємо GPIO-вихід з функціє. ШІМ.

STM32F103C8T6 має безліч інших функцій на інших ніжках, які також можна використовувати лише як виходи для прийому та відправлення сигналу. Ці ніжки будуть необхідними для прийняття сигналів датчиків струму у ланці.

Для отримування сигналу лічильнику, потрібна ніжка з можливість обробки переривання. Таку функцію забезпечують таймери загального призначення, які також здатні генерувати сигнали ШІМ.

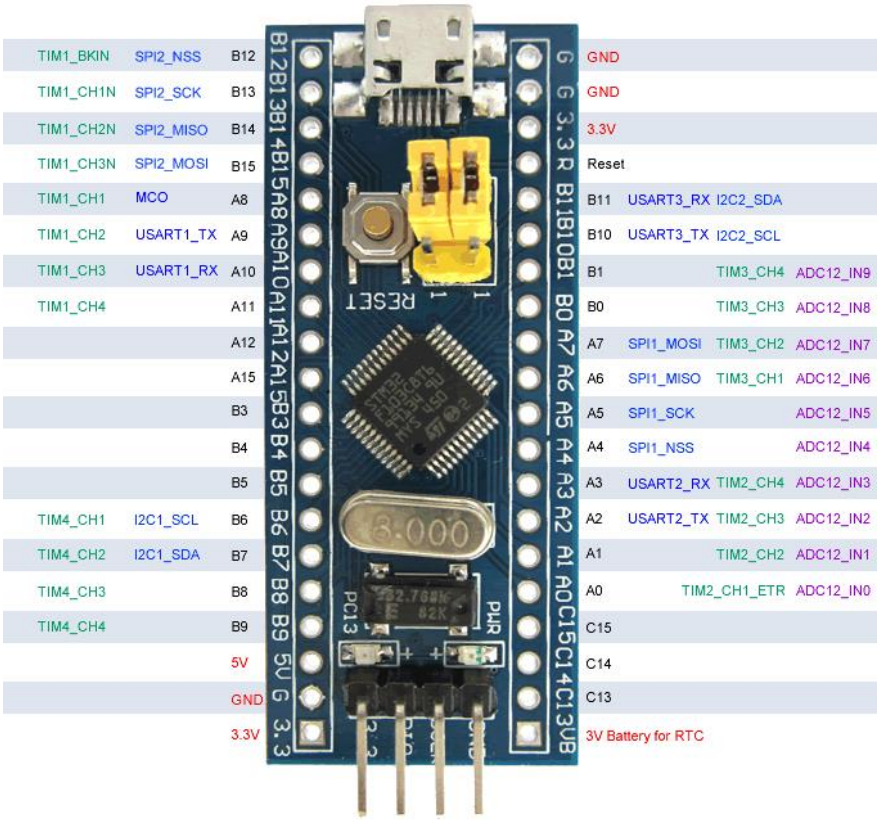


Рисунок 3.3 – Розпіновка мікроконтролеру STM32F103C8T6

Отже, мікроконтролер STM32F103C8T6 недорогий інструмент розробки, який має широкий спектр функціоналу. Він задовольняє всі технічні вимоги до мікроконтролеру та забезпечує усі потрібні можливості для ефективного використання ресурсів апаратно-програмного комплексу.

3.3 Програмування мікроконтролеру

Програмування та відладка мікроконтролеру даної моделі значно складніша, ніж у сімейства Arduiono. Для того, щоб завантажити готову, скомпільовану прошивку на мікроконтролер, потрібно підключення програматора. Для даної моделі був обраний програматор ST-Link V2, який відрізняється своєю дешевизною та оптимальною роботою.

Спочатку потрібно підключити програматор до мікроконтролера. Для такого підключення використовуються кабелі типу «мама-мама». Порядок підключення зображений на рисунку 3.4.

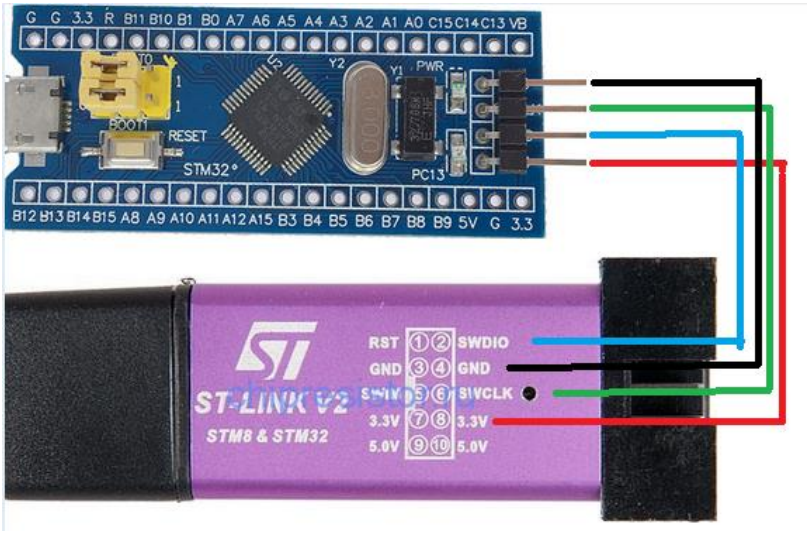


Рисунок 3.4 – Підключення програматора до мікроконтролеру

Після з’єднання відповідних ніжок, програматор підключається через USB-порт до комп’ютера, та, утримуючи кнопку перезавантаження (Reset) на мікроконтролері, запустити

3.4 Одноплатний комп’ютер Orange Pi

Для забезпечення передачі даних з пристрою на будь-який комп’ютер, було вирішено використовувати протокол передачі даних TCP/IP. Для збору даних та забезпечення їх передачі потрібно було обрати плату, яка відповідала б вимогам апаратно-програмного комплексу.

Після аналізу ринку, було визначено дві моделі, які мають потрібні характеристики: Raspberry Pi та Orange Pi. Це популярні моделі відносно недорогих одноплатних комп’ютерів, які забезпечують надійну роботу та легкі у програмуванні.

Orange Pi – одноплатний комп’ютер з відкритим вихідним кодом. Він може запускати Android 4.4, Ubuntu, Debian, Rasberry Pi Image, а також Banana Pi Image. Він використовує AllWinner H3 SoC, і має 1GB DDR3 SDRAM

Така плата має багато застосувань. На його основі можна побудувати як повністю функціональний комп’ютер, так і мережевий сервер. Він ідеально підходить для створення пристроїв різної складності та дозволяє у повному обсязі використовувати як і апаратні можливості, так і можливості невеликого комп’ютера.

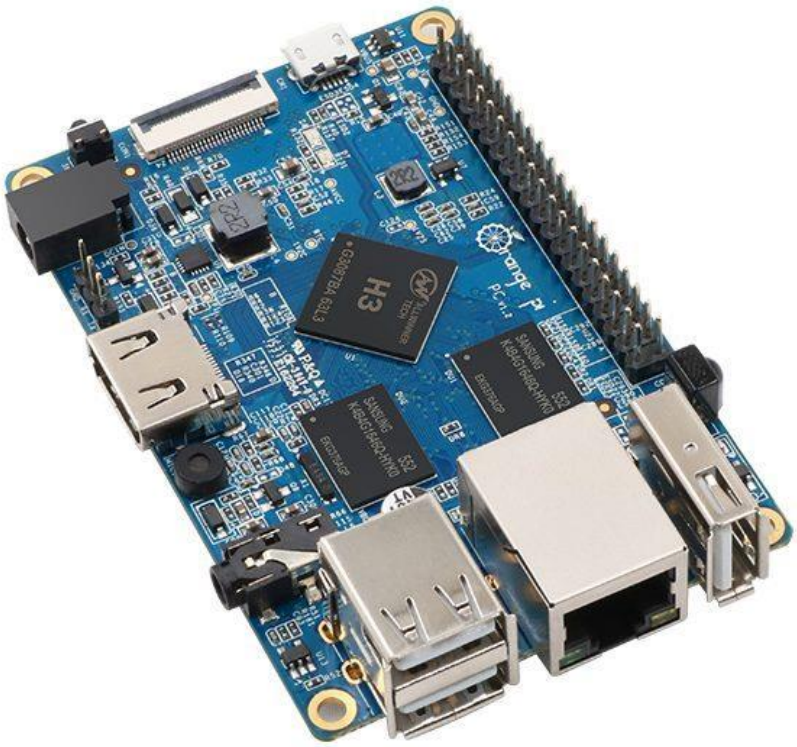


Рисунок 3.5 – Одноплатний комп’ютер Orange Pi

Дана модель має багато переваг. Одним із важливих пунктів є можливість встановлення операційних систем на плату. Вона пропонує широку лінійку операційних систем, кожна з яких має свої переваги та свої недоліки.

Операційна пам’ять комп’ютера дуже важлива для розробки, і ця плата забезпечує її великий об’єм – 1Гб. Оперативна пам’ять (оперативна пам’ять) - це апаратне забезпечення в обчислювальному пристрої, де операційна система (ОС), прикладні програми і дані в поточному використанні зберігаються, щоб їх можна було швидко отримати від процесора пристрою.

Оперативна пам’ять - це основна пам’ять комп’ютера, і її читання набагато швидше, ніж для інших типів сховища, наприклад жорсткого диска (HDD), твердотільного накопичувача (SSD) або оптичного приводу.

Оперативна пам’ять є нестабільною. Це означає, що дані зберігаються в оперативній пам’яті до тих пір, поки комп’ютер ввімкнено, але вони втрачаються під час вимкнення комп’ютера. Коли комп’ютер перезавантажується, ОС та інші файли перезавантажуються в ОЗУ, як правило, з жорсткого диска або SSD.

З-за своєї волатильності, пам’ять випадкового доступу не може зберігати постійні дані. Оперативну пам’ять можна порівняти з короткочасною пам’яттю людини, а жорсткий диск - до довготривалої пам’яті людини. Короткочасна пам’ять зосереджена на безпосередній роботі, але вона може зберігати лише обмежену кількість фактів у будь-який момент часу. Коли короткочасна пам’ять людини заповнюється, її можна оновити фактами, що зберігаються в довгостроковій пам’яті мозку.

Комп’ютер також працює таким чином. Якщо RAM заповнюється, процесор комп’ютера повинен неодноразово переходити на жорсткий диск, щоб перекривати старі дані в оперативній пам’яті з новими даними. Цей процес уповільнює роботу комп’ютера.

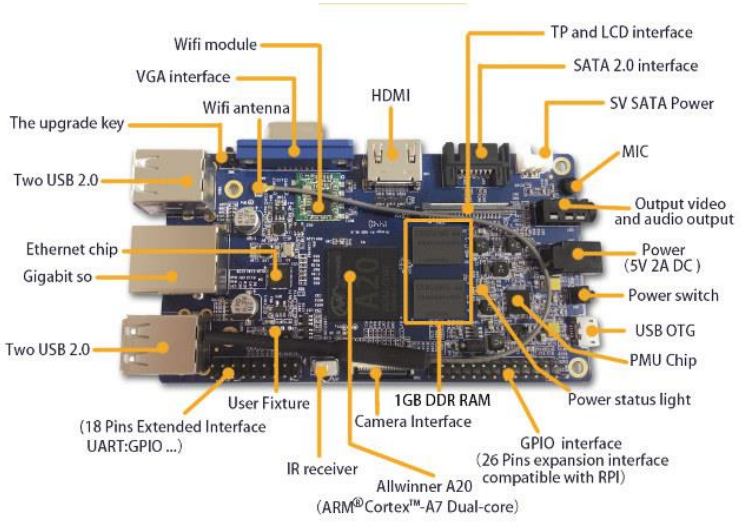


Рисунок 3.6 – Будова одноплатного комп’ютеру Orange Pi

Orange Pi є одним з найвідоміших брендів серед одноплатних комп’ютерів. Вартість Orange Pi починається від 15 доларів. Це один з найдешевших пристроїв з хорошими характеристиками.

Переваги Orange Pi:

— Низька вартість - плата Orange Pi дешевше, ніж Raspberry Pi приблизно в 2 рази;

— Різні моделі, які підходять для конкретних завдань;

— GPIO сумісність;

— Сумісність з іншими компонентами для комп’ютерів Orange;

— Висока швидкодія;

— Великий обсяг пам’яті;

— Наявність декількох USB роз’ємів (в залежності від моделі) і HDMI висновків;

— Велика кількість операційних систем;

— Непогана продуктивність.

Недоліки:

— Сильно нагрівається до 80 градусів, обов’язково потрібно купувати охолодження для радіатора;

— Не підходить для складних процесів;

— У керівництві написано, що потрібно блок живлення на 3 А;

— Високий відсоток браку.

— У комплектації є тільки сама плата і інструкція англійською мовою. До комп’ютера додатково потрібно буде докуповувати кабель і блок живлення.

— Використання одноплатних ПК

Одноплатні комп’ютери можуть використовуватися в різних проектах. Вони можуть використовуватися в ролі Smart-TV і мультимедіа. Плата дозволяє розширити функціонал за невисоку вартість. Також популярним застосування одноплатніков є ігрові консолі. Для цього є спеціальні операційні системи з емуляторами - Sega, PS 1, PSP і багато інших. Плати використовуються для пошуку інформації в інтернеті по попередньо встановленого браузеру Chromium, перегляду відеозаписів, роботи з офісними пакетами. Можна створити домашню хмара або веб-сервер на базі одноплатного комп’ютера. Користувач отримає тиху і ефективну систему з високою швидкодією. Використовувати їх можна в системах «розумний будинок».

Виробник Orange Pi рекомендує використовувати для приладів нового покоління блоки живлення з напругою 5 Вольт і струмом 3 Ампер. Це повинно виключити проблеми з запуском і гарантує стабільну роботу приладу. Але за фактом мікрокомп’ютер споживає не більше 1 А навіть при підключеної периферії.

Блок живлення приєднується до Orange Pi через роз’єм 4x1,7 мм. Для підключення потрібно спеціальний кабель живлення. Також харчування можна підключити через Піни GRIO: плюс до висновку 5V, а мінус - до землі. Перед підключенням важливо звіритися з документацією.

Харчування можна подавати і через Power Bank. Важливо купити якісний переносний акумулятор. Не всі дешеві пристрої можуть одночасно заряджатися і пропускати через себе заряд для живлення Orange Pi.

В першу чергу, обидва пристрої відрізняються своєю вартістю. Orange Pi дешевше, ніж Raspberry Pi приблизно в 1,5-2 рази. Це пов'язано з тим, що в Orange Pi використовується дешевший процесор. Доведеться купувати систему охолодження. Використання таких приладів призводить до того, що Orange Pi може нагріватися, а зазначена виробником частота 1,6 ГГц не відповідає дійсності. Справжнє значення частоти знаходиться на рівні 1,2 ГГц. Незважаючи на ціну, процесор є потужним і відрізняється високою продуктивністю.

На вартість впливає і бренд. Raspberry Pi є основоположником напрямку одноплатних комп'ютерів, тому і ціна подібних виробів вище. При виникненні проблем вирішувати їх доведеться самостійно, в той час як про неполадки і їх виправлення з Raspberry Pi можна прочитати на форумах.

Лінійка пристроїв Orange Pi досить обширна. Користувач може знайти пристрій на будь-яке завдання. Китайська фірма пропонує більше десяти різних моделей зі своїми відмітними технічними характеристиками.

Також важливою відмінністю є кількість підтримуваних операційних систем. Пристрої Orange Pi підтримують велику кількість ОС. На офіційному сайті виробника можна переглянути список і перевірити, які системи підходять для тієї чи іншої версії плати.

У користувачів є претензії до якості збірки Orange Pi. Пайка хоч і акуратна, але можуть бути не видалені залишки флюсу. Також деякі конектори розташовані незручно - наприклад, при приміщенні плати в корпус користувач ризикує залишитися без зручного доступу до GRIO висновків.

Плати Orange Pi не зможуть використовуватися в ресурсномістких проектах. Але вони відмінно підходять для виконання нескладних функцій - наприклад, для головного пристрою для управління розумним будинком, для принт-сервера. Це хороші пристрої для новачків. Для складних процесів краще використовувати Raspberry Pi.

3.5 Програмування одноплатного комп’ютеру

Для зручної розробки та компіляції на плату Orange Pi потрібно становити операційну систему. Orange Pi підтримує більше 15 операційних систем:

Для розробки була обрана Armbian.

Armbian - це платформа базової операційної системи для одноплатних комп’ютерів, на яку можуть довіряти інші проекти. Це легкий дистрибутив Debian або Ubuntu, що спеціалізується на розробці плат ARM. Кожну систему компілюють, збирають і оптимізують Armbian Build Tools. Він має потужні засоби побудови та розробки програмного забезпечення для створення власних збірок.

Головними перевагами для використання у проекті:

- Офіційна підтримка плати Orange Pi, оскільки це гарантує надійну роботу системи.
- Базується на UBUNTU16
- Для системи існує широкий вибір забезпечення розробника
- Зручна конфігурація апаратної частини Orange Pi
- Адаптер Ethernet з сервером DHCP і SSH готовий за замовчуванням (22)
- Бездротовий адаптер з DHCP готовий, якщо він присутній, але вимкнено. Ви можете використовувати armbian-config для підключення до маршрутизатора або створення AP
- Додано скрипт NAND, SATA, eMMC і USB (nand-sata-install)
- Оновлення здійснюються за допомогою стандартного методу оновлення apt-get
- Скрипт входу показує: назва плати з великим текстом, база розсилки, версія ядра, завантаження системи, час роботи, пам’ять, IP-адреса, темп процесора, темп диска, температура навколишнього середовища від температури, якщо виходи, використання картки SD, умови акумулятора та кількість оновлення для встановлення

4 ЗАСОБИ РОЗРОБКИ

Для розробки даного програмного продукту використовувалось багато різних технологій для роботи з мікроконтролером, одноплатним комп’ютером та для створення клієнтського додатку. Було проаналізовано вимоги кожного етапу розробки та вибрано наступні засоби розробки програмного забезпечення.

4.1 Інструмент графічної конфігурації мікроконтролера STM32CubeMx

STM32CubeMX - це графічний інструмент, що дозволяє дуже легко конфігурувати мікроконтролери STM32 і мікропроцесори, а також генерувати відповідний код ініціалізації для ядра Arm® Cortex®-M або часткового дерева пристроїв Linux® для Arm® Cortex®- Ядро, через покроковий процес [13].

Ключовими перевагами використання даного продукту є:

- Інтуїтивний мікроконтролер STM32 і мікропроцесорний вибір
- Багатий простий у використанні графічний інтерфейс користувача, що дозволяє налаштувати:
- Закріплення з автоматичним вирішенням конфліктів
- Периферійні та проміжні функціональні режими з динамічною перевіркою обмежень параметрів для ядра Arm® Cortex®-M
- Дерево годинника з динамічною перевіркою конфігурації
- Послідовність живлення з розрахунковими результатами споживання
- Генерація проекту кодування ініціалізації, сумісного з компіляторами IAR ™, Keil® та GCC, для ядра Arm® Cortex®-M
- Генерація часткового дерева пристроїв Linux® для ядра Arm® Cortex®-A (мікропроцесори STM32)
- Наявність як окремого програмного забезпечення, що працює під керуванням Windows®, Linux® та macOS® (MacOS є торговою маркою компанії Apple Inc., зареєстрованою в США та інших країнах.) Операційних систем, або за допомогою плагіна Eclipse

4.2 UART-зв’язок

UART (Універсальний асинхронний приймач / передавач) - це мікрочіп з програмуванням, який керує інтерфейсом комп'ютера до його приєднаних послідовних пристроїв. Зокрема, він надає комп'ютеру інтерфейс RS-232C Data Terminal Equipment (DTE), щоб він міг «розмовляти» та обмінюватися даними з модемами та іншими послідовними пристроями. У рамках цього інтерфейсу UART також:

- Перетворює байти, які він отримує від комп'ютера вздовж паралельних ланцюгів, в один потік послідовного біта для вихідної передачі
- На вхідній передачі, перетворює послідовний потік бітів в байти, які обробляє комп'ютер
- Додає біт парності (якщо він був обраний) на вихідні передачі і перевіряє парність вхідних байтів (якщо вибрано) і відкидає біт парності
- Додає початкові та кінцеві позначки на вихідні і видаляє їх з вхідних передач
- Ручки переривання з клавіатури та миші (які є послідовними пристроями зі спеціальними портами s)
- Може обробляти інші види переривань і керування пристроями, які вимагають координації швидкості роботи комп'ютера зі швидкістю пристрою
- Більш просунуті UART забезпечують певну кількість буферизації даних, щоб потоки даних комп'ютерів і послідовних пристроїв залишалися координованими. Останній UART, 16550, має 16-байтовий буфер, який може бути заповнений до того, як процесор комп'ютера повинен обробляти дані. Оригінальний UART був 8250. Якщо ви купуєте внутрішній модем сьогодні, це, ймовірно, включає в себе 16550 UART (хоча ви повинні запитати, коли ви купуєте його). За даними виробника модемів US Robotics, зовнішні модеми не містять UART. Якщо у вас

старий комп'ютер, можна додати внутрішній 16550, щоб отримати максимальну віддачу від зовнішнього модему.

Такий зв’язок є надійним та достатньо швидким, тому йогодоцільно використовувати для зв’язку апаратних частин системи.

Приклад з’єднання ніжок з UART-зв’язком показано на рисунку 4.1. Tx – ніжка передачі даних, Rx – ніжка прийому даних.

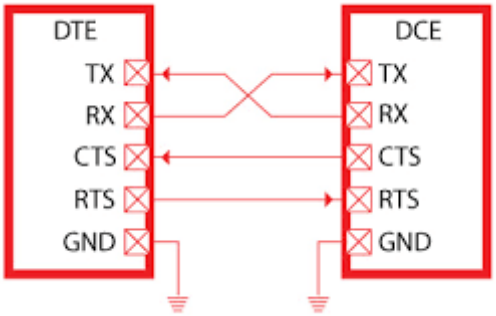


Рисунок 4.1 – З’єднання девайсів через Uart ніжки

4.3 TCP/IP-з’єднання

TCP/IP або протокол керування передачею через Інтернет-протокол - це набір протоколів зв'язку, які використовуються для з'єднання мережевих пристроїв в Інтернеті. TCP / IP також може використовуватися як протокол зв'язку в приватній мережі (інтранет або екстранет).

Весь набір інтернет-протоколів - набір правил і процедур - зазвичай називають TCP / IP, хоча інші входять до комплекту.

TCP / IP визначає, як обмінюються даними через Інтернет, надаючи комплексні комунікації, які визначають, як вони повинні бути розбиті на пакети, адресовані, передані, маршрутизовані та отримані в пункті призначення. TCP/IP вимагає невеликого центрального управління, і він розроблений для того, щоб зробити мережі надійними, з можливістю автоматично відновлюватися від збою будь-якого пристрою в мережі.

Два основні протоколи в наборі інтернет-протоколу служать певним функціям. TCP визначає, як програми можуть створювати канали зв'язку через мережу. Він

також керує тим, як повідомлення збирається в менші пакети, перш ніж вони потім передаються через Інтернет і знову збираються в правильному порядку на адресу призначення.

IP визначає, як адресувати і направляти кожен пакет, щоб переконатися, що він досягає потрібного місця призначення. Кожен комп'ютер шлюзу в мережі перевіряє цю IP-адресу, щоб визначити, куди переслати повідомлення.

4.4 Мови програмування C, C++, C#

Для розробки прошивки мікроконтролеру, було обрано мову С.

Мова С є універсальною мовою програмування. Він тісно пов’язаний з операційною системою UNIX, так як був розвинений на цій системі і так як UNIX і її програмне забезпечення написано на С. Сама мова, проте, не пов’язаний з якоюсь однією операційною системою або машиною; і хоча його називають мовою системного програмування, так як він зручний для написання операційних систем, він з рівним успіхом використовувався при написанні великих обчислювальних програм, програм для обробки текстів і баз даних.

Мова С - це мова щодо «низького рівня». У такій характеристиці немає нічого образливого; це просто означає, що С має справу з об’єктами того ж виду, що й більшість ЕОМ, а саме, з символами, числами і адресами. Вони можуть об’єднуватися і пересилатися за допомогою звичайних арифметичних і логічних операцій, здійснюваних реальними ЕОМ.

Мова С не є мовою до строгих типами в сенсі Паскаля або Алгола-68. Він порівняно поблажливий до перетворення даних, хоча і не буде автоматично перетворювати типи даних з буйною невимушеністю мови PL / 1. Існуючі компілятори не передбачають ніякої перевірки під час виконання програми індексів масивів, типів аргументів і т.д.

У мові С є покажчики і можливість адресної арифметики. Аргументи передаються функцій за допомогою копіювання значення аргументу, і викликана функція не може змінити фактичний аргумент на що викликає програмі.

Якщо бажано добитися «виклику за посиланням», можна неявно передати покажчик, і функція зможе змінити об’єкт, на який цей покажчик вказує. Імена масивів передаються зазначенням початку масивів, так що аргументи на кшталт масивів ефективно викликаються за посиланням.

До будь-якої функції можна звертатися рекурсивно, і її локальні змінні зазвичай «автоматичні», тобто Створюються заново при кожному зверненні. Опис однієї функції не може міститися всередині іншого, але змінні можуть описуватися відповідно до звичайної блокової структурою. Функції в програмі можуть транслюватися окремо. змінні по відношенню до функції можуть бути внутрішніми, зовнішніми, але відомими тільки в межах одного вихідного файлу, або повністю глобальними. внутрішні змінні можуть бути автоматичними або статичними. Автоматичні змінні для більшої ефективності можна поміщати в регістри, але оголошення регістра є тільки вказівкою для компілятора і ніяк не пов’язане з конкретними машинними регістрами.

Плюси мови С:

1. Портативна мова: Програми С, написані на одному комп’ютері, можуть працювати на будь-якому комп’ютері без будь-яких змін програмного коду або з невеликими змінами.

2. Створення блоку для інших мов: Програма С виступає в якості будівельного блоку для інших мов програмування. Програми, написані на С, є більш ефективними і легкими для розуміння. Основні принципи мови С застосовуються на інших мовах.

3. Структурована мова програмування: Програма С - це орієнтована на процедури мова з набором функціональних модулів і блоків, які утворюють повну програму. Структуровані блоки спрощують налагодження, тестування та підтримку програми.

4. Легко навчитися: Вивчити мову С дуже легко і виступає основою для розуміння інших складних мов. Для легкого розуміння він використовує синтаксис, схожий на англійську мову.

5. Вбудована функція: мова С дає вам можливість використовувати кілька вбудованих функцій у бібліотеці С для розробки програми.

6. Визначена користувачем функція: Крім стандартної функції в бібліотеці С, ви можете створити власну функцію, визначену користувачем, для вирішення конкретної проблеми. Додавання додаткових функцій до бібліотеки С полегшує роботу програм.

7. Досліджуйте приховані об’єкти: У програмі С ви можете легко отримати доступ до прихованих або заблокованих об’єктів від використання іншими мовами програмування.

8. Програми прискорення: оскільки інші мови засновані на програмі с, це прискорює програму, розроблену з використанням інших мов програмування.

Програми на С працюють швидше, ніж на інших мовах.

9. Мова компіляції: Вона дозволяє ущільнити програмний код у виконувану інструкцію, а не перекладати перекладачами.

10. Низький рівень абстракції: мова програмування С близька до специфікацій системного обладнання, що дозволяє легко знати, як працює мова більш високого рівня і взаємодіяти з машиною.

Мінуси:

1. Безпека даних: Існує багато переповнення буфера на мові Сі, і це може призвести до перезапису інформації в пам’яті. Коли покажчики оновлюються неправильними даними, це призведе до пошкодження пам’яті.

2. Відсутність перевірки часу виконання: мова С не дозволяє перевіряти час виконання, що ускладнює виправлення помилок, якщо ви продовжите програму. В основному це виконує компіляцію перевірки типу.

3. Немає суворої перевірки типу: при передачі даних до параметрів, немає суворої перевірки типу даних, оскільки ми можемо передавати ціле значення параметру. Підтвердження правильного типу даних не використовується.

4. Немає повторного використання коду: мова С не має функцій ООП, які підтримують повторне використання вихідного коду. Він не підтримує конструкторів і деструкторів.

5. Концепція простору імен: мова С не підтримує простір імен програми, тому неможливо одночасно оголосити дві змінні, як у програмі С ++.

6. Немає концепцій ООП: Об'єктно-орієнтовані концепції програмування, такі як абстракція даних, поліморфізм, успадкування та інші концепції програмування С ++, не підтримуються мовою С. Кожен алгоритм в С - це набір викликів функцій.

7. Вплив на сьогоднішнє програмування: Програма С не підтримує достатньо бібліотечних функцій, які можна використовувати для обробки складного середовища програмування сьогодні.

8. Проблеми реального світу: він не може бути використаний для вирішення проблем реального програмування.

9. Розширення питань програми: Якщо ви продовжите програму, вам буде дуже важко виправити помилки та помилки. Мова С ефективний при роботі з простими проектами.

10. Конструкції високого рівня: Необхідно вручну створити конструкції високого рівня в мові програмування С. Перед використанням треба налаштувати бібліотеки третіх сторін та інші рішення.

Головною перевагою для використання С є широкий бібліотек для взаємодії з контролерами та забезпечує надійну роботу з пам'яттю пристрою.

Для розробки серверної частини було обрано мову С++.

С++ - мова програмування високого рівня, розроблена Б'ярном Страуструпом у 1979 році. Мова вже більше 20 років активно використовується у розробці програмного забезпечення, зокрема, для системного програмування, написання драйверів, потужних серверних і клієнтських програм, а також для розробки движків для відеоігор.[9]

Створюючи мову, автори намагались зберегти сумісність з мовою С так, щоб більшість програм, можна було збирати з допомогою компілятора С++. Обидві мови мають схожий синтаксис, що дозволяє легко вивчити мову С++, якщо попередньо знайомий з мовою С.

Плюси мови.

Широка підтримка: С ++ є дуже популярною мовою, і тому багато компіляторів і бібліотек вже існують, які сумісні з нею. Таким чином, набагато простіше розробляти великі проекти з багатьма залежностями, ніж з більш езотеричними мовами, які мають менше підтримки. Крім того, вона досить популярна, що дуже ймовірно, що код був написаний раніше для будь-якої програми, яку можна собі уявити, так що легко взяти чужий код і переробити його, щоб задовольнити індивідуальні потреби, а не написати все з нуля. Є також багато бібліотек з відкритим кодом для С ++, які можна знайти в Інтернеті, дозволяючи програмістам використовувати переписані бібліотеки замість того, щоб писати свій код або переробляти чужий код. Нарешті, існує також компілятор для кожної великої операційної системи, тому речі, розроблені в С ++, неймовірно портативні.

Потужність: Оскільки С ++ не вимагає встановлення спеціальної середовища виконання для її запуску, можна створити будь-яку програму, аж до програмування на низькому рівні до складних графічних інтерфейсів.

Швидкість: Оскільки вона компілюється, С ++ отримує велику швидкість. Необроблені мови повинні інтерпретуватися під час виконання, тобто кожна дія - це 2-х етапний процес. Скомпільовані мови попередньо інтерпретуються так, що кожен крок у виконуваному файлі є однією машинною інструкцією.

Подібність до інших мов: Багато інших мов, таких як С, С # і Java, мають дуже схожі синтаксиси до С ++, що робить їх легкими для вивчення для тих, хто вже знає С ++.

Невелика стандартна бібліотека: стандартна бібліотека С ++ невелика в порівнянні з іншими мовами, такими як Java, що дозволяє програмісту робити більше з меншими обмеженнями.

Проте мова має також багато мінусів, які потрібно враховувати при виборі мови.

Мова небезпечна: стандарт дозволяє багато речей, які можуть призвести до несподіваної поведінки. Це дозволяє програмісту більше можливостей розробки, але також змушує програміста робити більше перевірок. Вона не виконує перевірки кордонів на масивах і дозволяє неналежне перетворення типу, що може призвести до пошкодження пам'яті. Це проблема, яку дуже важко налагодити.

Мало управління пам'яттю: C++ майже не управляє пам’яттю самостійно. Тому при розробці потрібно активно слідкувати за використанням пам’яті та її вчасним звільненням.

Архаїчна орієнтація об'єктів: Система об'єктної орієнтації в C ++ необґрунтовано основна в порівнянні з іншими мовами. Інші мови, такі як Java, мають набагато кращі та гнучкіші системи.

Відсутність користувацьких операторів: багато мов дозволяють програмістам визначати власні оператори. Наприклад, я можу визначити, що<-> b буде міняти дані в a з даними в b. Можна перевизначити існуючі оператори в C ++, але створення нових неможливе. Необхідна функціональність повинна бути виконана з функціями, які можуть зменшити читаність коду.

Відсутність алгебраїчних типів даних: Прості типи, такі як кортежі, не включаються, що змушує програмістів робити власні реалізації або знаходити бібліотеку, якщо вони хочуть їх використовувати.

Функції не є типами першого класу: Перші типи класів - це типи, які можуть бути повернуті з функцій, можуть бути передані функціям і можуть бути побудовані під час виконання . Ці типи не враховуються, тобто деякі функції програмування, які будуть доступні на інших мовах, недоступні в C++. Обхідні шляхи можуть бути створені за допомогою вказівників на функції, але це збільшує складність коду, що ускладнює запис і читання.

Дещо складний синтаксис, яким повинні користуватися користувачі C++.

Складний: Синтаксис складний і стандартна бібліотека невелика, що робить C++ важкою для вивчення для тих, хто має невеликий досвід програмування.

Синтаксично строгий: для синтаксису не так багато гнучкості, тому написання коду може бути важким для читання.

Нестандартизовані функції вищого рівня: Загальні функції програми, такі як графічні інтерфейси, мережеві мережі та потоки, залежать від операційної системи, що змушує програмістів або зробити декілька версій програми, або включати зовнішні бібліотеки, які вже зробили це. Найновіший стандарт додав деяку стандартизацію для потоків, але він має ще довгий шлях порівняно з мовами, подібними до Java.

Для розробки інтерфейс-додатку було обрано мову C#.

Мова C# - об’єктно-орієнтована мова, яка має багато інструментів для створення інтерфейсів. Зокрема ця мова підтримує інструмент розробки Windows Forms, який дозволяє розробляти зручний та функціональний інтерфейс, який буде інтуїтивним, тобто зрозумілим для користувача. Windows Forms має широкий набір елементів управління, які можна додавати на форми: книпки, текстові поля, поля для введення, вкладки, графіки і т. д. Список усіх елементів управління представлений у розділі «Інструменти». Також даний інструмент дозволяє створювати власні елементи управління, що значно розширяє можливості створення інтрефейсу.

Також у роботі використовувались стандартні бібліотеки для організування мережевого зв’язку – System.Net та System.Net.Sockets

4.5 Бібліотека HAL

Для безпосередньо роботи з контролером використовувалась бібліотека HAL. HAL-драйвери орієнтовані не на периферійні пристрої мікроконтролера, а на функціональні можливості цих пристроїв. Наприклад, на одному і тому ж таймері можуть бути реалізовані: відлік інтервалів часу, циклічне переривання, ШІМ, захоплення і т.п. Для всіх цих функцій існують різні драйвери HAL. Більш того, один драйвер часто використовує кілька периферійних пристроїв. Наприклад, разом з тим же таймером HAL-драйвер може управляти і системою переривань. Як наслідок,

HAL-драйверами можна користуватися, не вникаючи в деталі роботи периферійних пристроїв. Вони значно полегшують працю розробника.

4.6 Бібліотека Wiring Op

WiringPi - це бібліотека доступу GPIO на основі GPIO, написана на мові C для пристроїв BCM2835, BCM2836 і BCM2837 SoC, що використовуються у всьому Raspberry Pi. версії. Випускається під ліцензією GNU LGPLv3 і може використовуватися з C, C ++ і RTB (BASIC), а також з багатьма іншими мовами з відповідними обгортками призначений для використання досвідченими C / C ++ програмістами.

WiringPi включає утиліту командного рядка gpio, яку можна використовувати для програмування та налаштування контактів GPIO. Можна використовувати це для читання і запису контактів і навіть використовувати його для керування ними зі скриптів оболонки.

WiringPi є розширюваним, і модулі надаються для розширення wiringPi для використання аналогових інтерфейсних пристроїв на Gertboard, а також для використання популярних мікросхем розширення MCP23x17 / MCP23x08 (I2C 7 SPI) GPIO, а також модуля, що дозволяє блокам до 4 595 регістрів зсуву з’єднуються разом для отримання додаткових 32-бітних результатів у вигляді однієї одиниці. Один з модулів розширення дозволяє використовувати ATmega (наприклад, Arduino або Gertboard) як більше розширення GPIO - через послідовний порт Pi.

Крім того, можете легко написати свої власні модулі розширення, щоб інтегрувати власні периферійні пристрої з wiringPi, якщо потрібно.

5 ІНТЕРФЕЙС КОРИСТУВАЧА

Користувацький інтерфейс включає в себе 3 вкладки, кожна з яких відповідає за свою функцію. У вкладці під назвою «Stats» (рисунок 5.1), відображається поточний стан електромережі. На схемі системи, біля точок, де мають знаходитись датчики струму, знаходяться чотирикутники, які відображають поточне значення датчиків: якщо чотирикутник має червону рамку, датчик показує відсутність струму на цій ділянці. Якщо чотирикутник набуває яскраво зеленого кольору, датчик фіксує струм на цій ділянці.

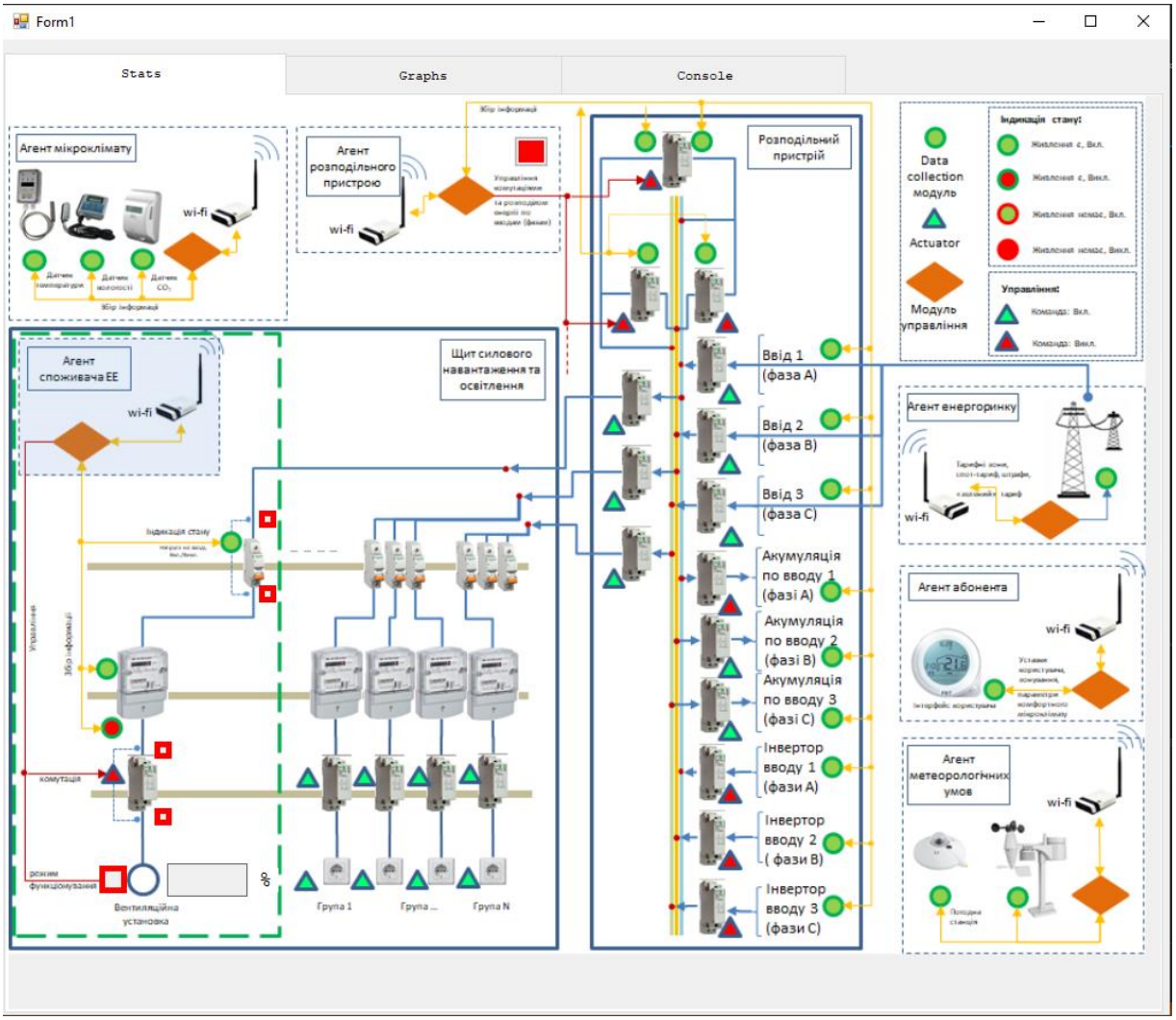


Рисунок 5.1 – Вкладка «Stats»

Коло позначки вентиляційної установки також міститься чотирикутний індикатор, який відображає чи ввімкнена в даний момент установка. З іншого боку

позначки міститься текстове поле, яке відображає поточне значення струму (у відсотках), який подається на вентиляційну установку.

Наступна вкладка – «Graph» – містить графік навантаженості електромережі. Графік змінюється у реальному часі та відображає показники лічильника кожну секунду. На рисунку 5.2 показано приклад роботи графіку.

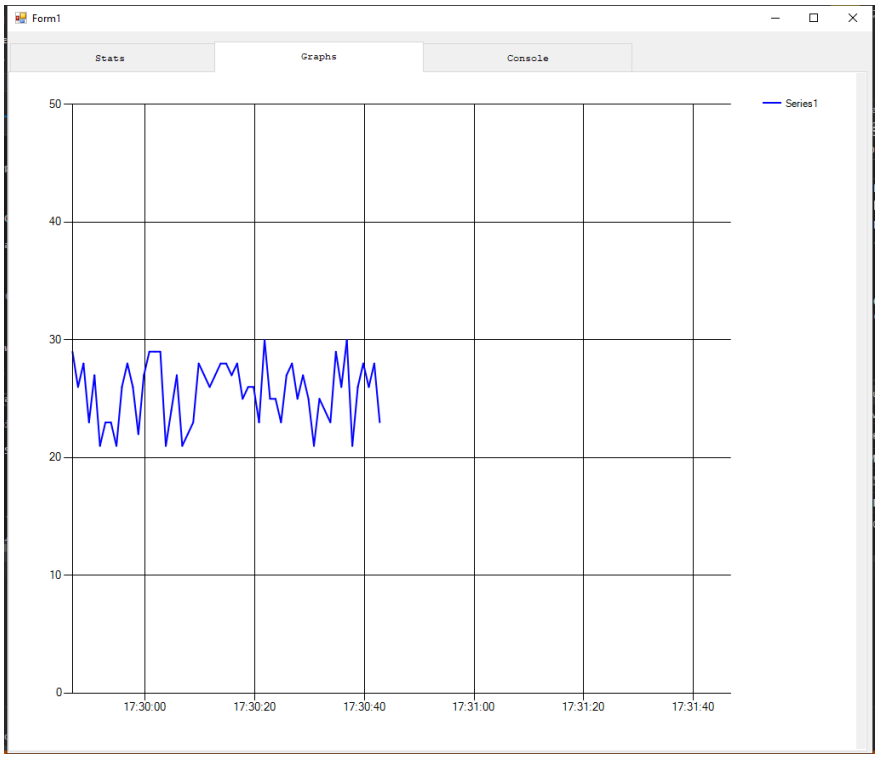


Рисунок 5.2 – Вкладка графіку

Третя вкладка має назву «Console» та є вкладкою тестування роботи агенту. Вкладка складається трьох блоків: блок ручного управління, блок автоматичного управління та консоль. На рисунку 5.3 показаний вигляд консольної вкладки.

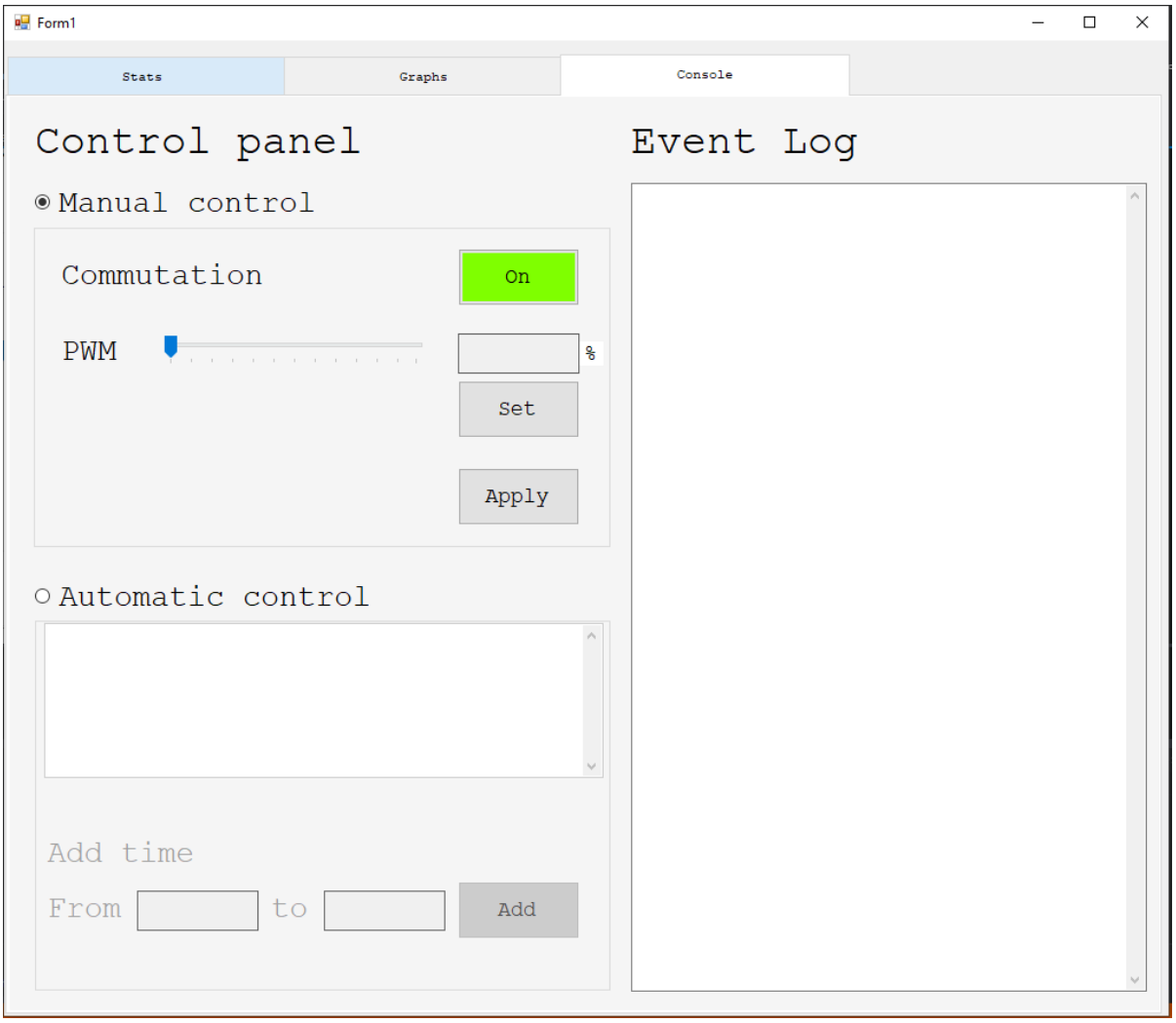


Рисунок 5.3 – Консольна вкладка

Консоль (Event log) записує усі зміни у системі та фіксує помилки. З її допомогою можна зручно відслідковувати події системи та знаходити помилку роботи.

Блоки автоматичного та ручного керування не можуть працювати одночасно, тому при виборі того чи іншого блоку,елементи другого автоматично блокуються до вибору цього блоку.

У блоці ручного керування можна ввімкнути та вимкнути установку, а також задати потужність сигналу установки. Для відправки команди агенту, потрібно натиснути кнопку «». Програма зафіксує значення, які передаватиме агенту, та відправить їх о TCP/IP-зв’язку.

Рисунок 5.4 наглядно показує роботу блоку ручного керування.

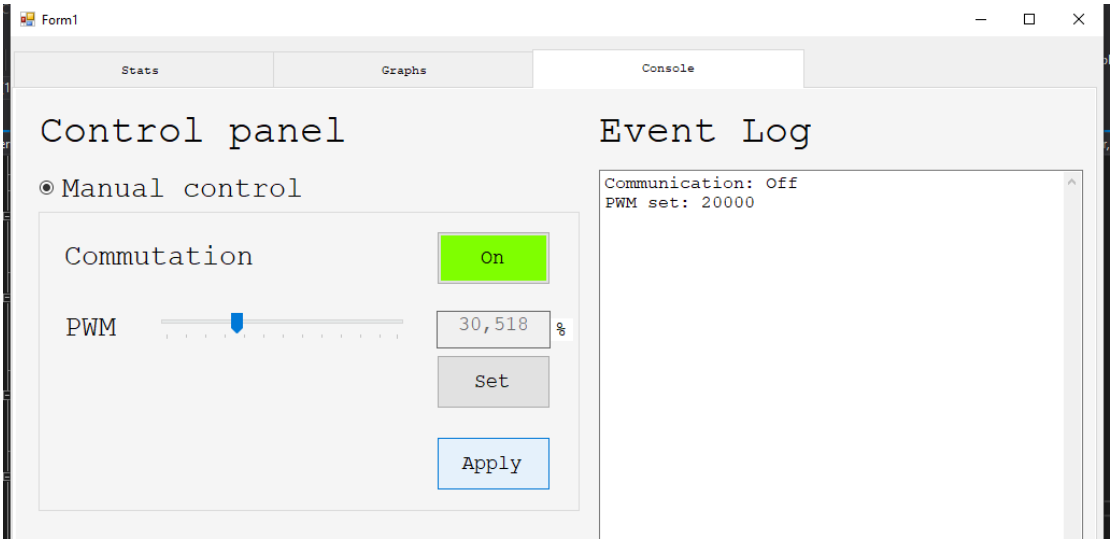


Рисунок 5.4 – Ручний контроль

Якщо клієнт не зміг зв’язатись з агентом, у консолі фіксується помилка та буде описана її причина. Уданому прикладі, клієнт намагався з’єднатись з агентом, який відключений від мережі, а тому не відкрив серверу для прийому. На рисунку 5.5 можна побачити описання помилки у консолі.

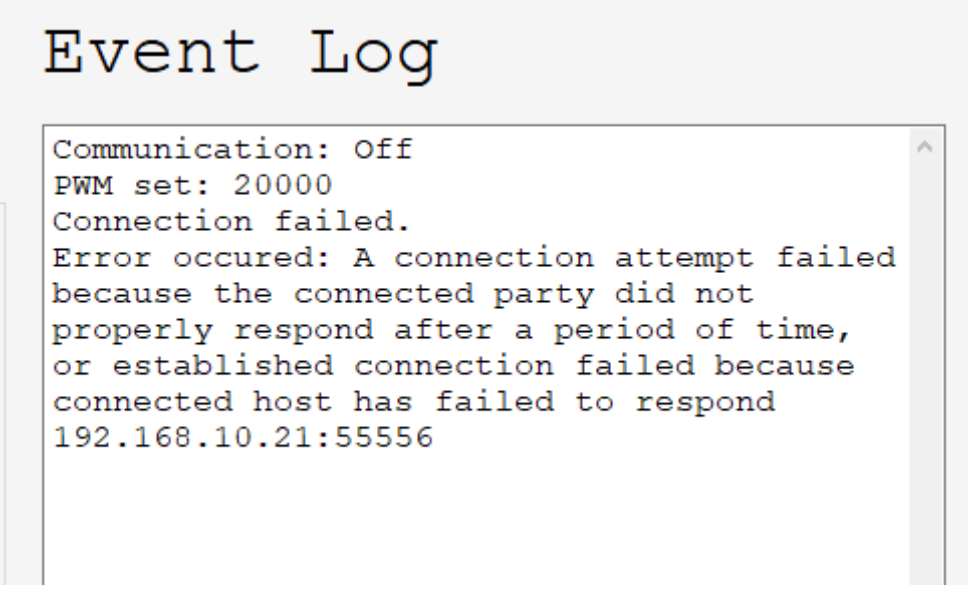


Рисунок 5.5 – Опрацювання помилок

Інший блок управління системою – блок автоматичного управління (рис 5.6). У цьому блоці можна задати час, у який буде запущена вентиляційна установка. Для

додання нового часу нижче від списку часових інтервалів міститься спеціальна форма, яка перевіряє правильність введеного часу.

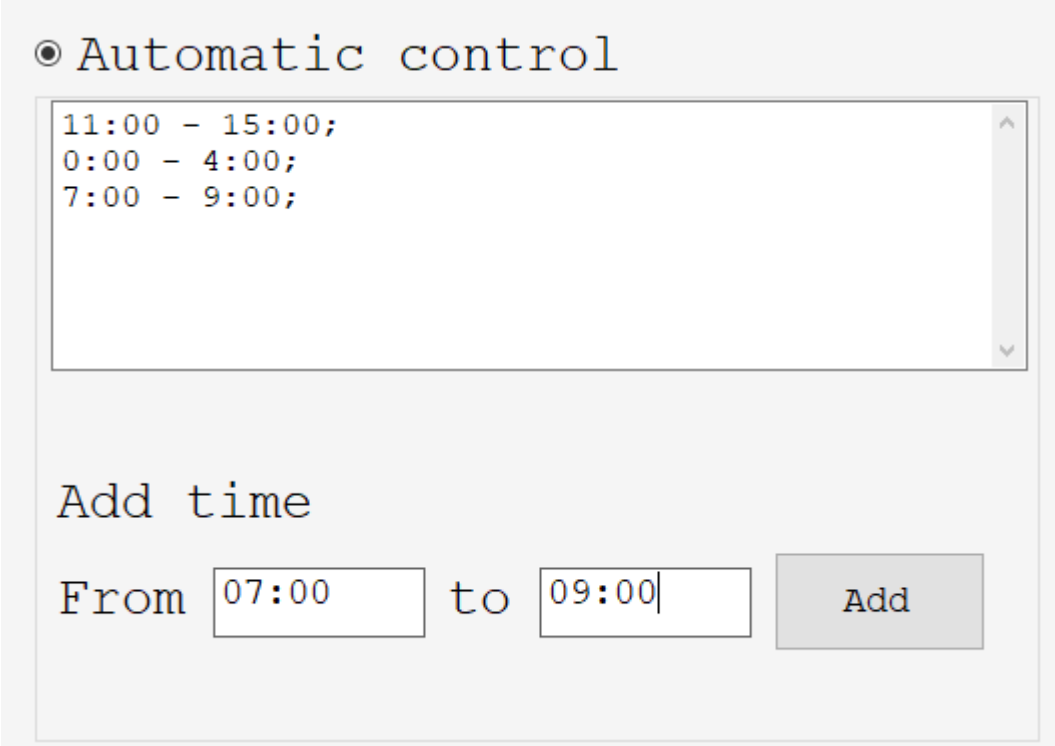


Рисунок 5.6 – Автоматичний контроль

Для виключення небажаного часового інтервалу, потрібно видалити відповідний рядок у списку часових інтервалів.

Після натиснення кнопки додавання часу, інтервал запишеться до списку інтервалів та відобразиться у вікні часових інтервалів та у консолі з повідомленням про додання інтервалу (рис. 5.7).

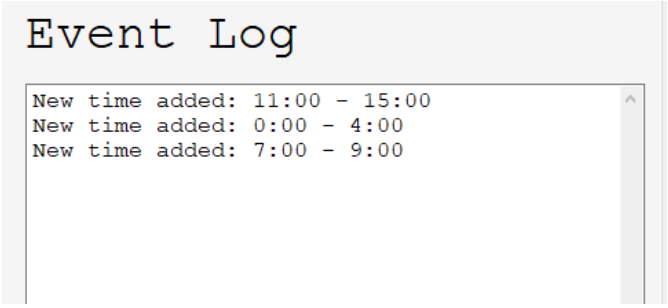


Рисунок 5.7 – Відображення змін у консолі

У випадку помилкового вводу часу, з’явиться вікно-повідомлення, яке опише характер помилки. Час не збережеться, форма чекатиме повторної спроби ввести час.

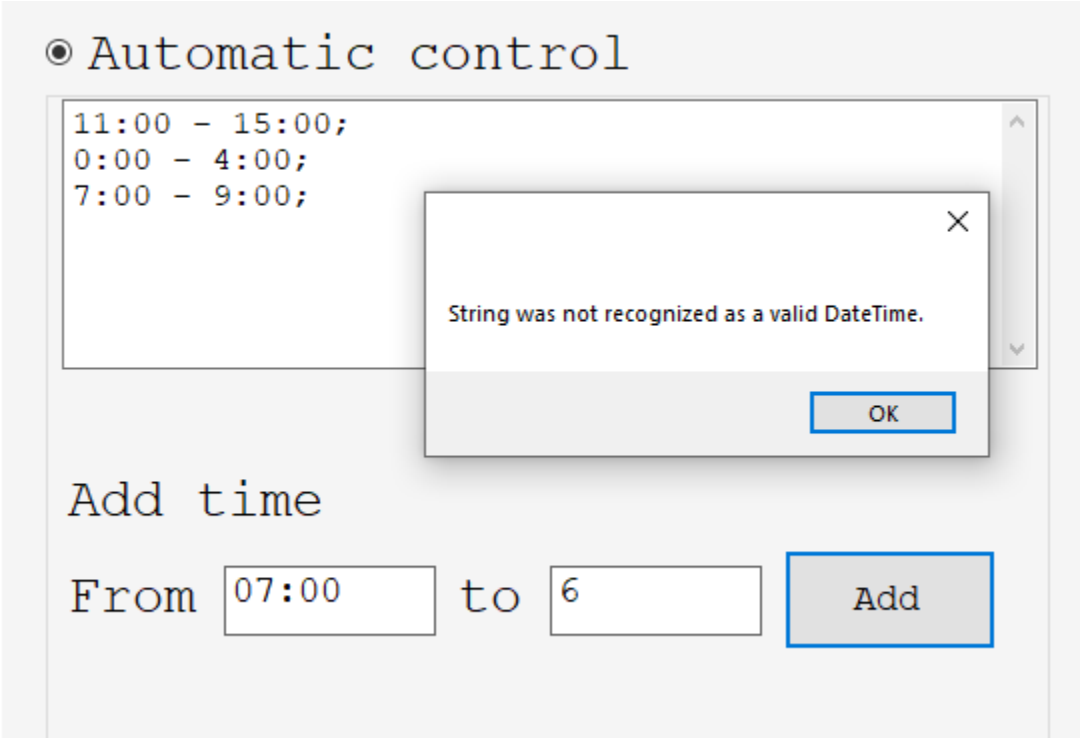
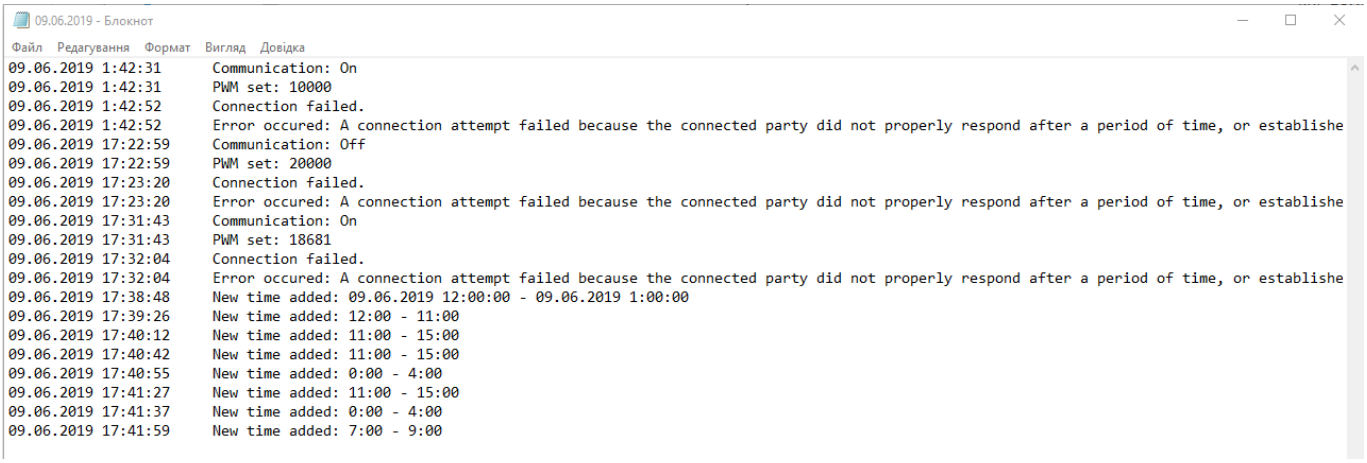


Рисунок 5.8 – Неправильний ввід часу

Для фіксування помилок, на випадок непередбачуваної зупинки програми, програма створює файл, ім’я якого, відображає дату запуску програми. Кожен крок програми записується навпроти часової мітки. Таким чином можна відслідковувати причину передчасного зупинки роботи програми.


09.06.2019 - Блокнот

Файл Редагування Формат Вигляд Довідка

```
09.06.2019 1:42:31      Communication: On
09.06.2019 1:42:31      PWM set: 10000
09.06.2019 1:42:52      Connection failed.
09.06.2019 1:42:52      Error occured: A connection attempt failed because the connected party did not properly respond after a period of time, or establishe
09.06.2019 17:22:59     Communication: Off
09.06.2019 17:22:59     PWM set: 20000
09.06.2019 17:23:20     Connection failed.
09.06.2019 17:23:20     Error occured: A connection attempt failed because the connected party did not properly respond after a period of time, or establishe
09.06.2019 17:31:43     Communication: On
09.06.2019 17:31:43     PWM set: 18681
09.06.2019 17:32:04     Connection failed.
09.06.2019 17:32:04     Error occured: A connection attempt failed because the connected party did not properly respond after a period of time, or establishe
09.06.2019 17:38:48     New time added: 09.06.2019 12:00:00 - 09.06.2019 1:00:00
09.06.2019 17:39:26     New time added: 12:00 - 11:00
09.06.2019 17:40:12     New time added: 11:00 - 15:00
09.06.2019 17:40:42     New time added: 11:00 - 15:00
09.06.2019 17:40:55     New time added: 0:00 - 4:00
09.06.2019 17:41:27     New time added: 11:00 - 15:00
09.06.2019 17:41:37     New time added: 0:00 - 4:00
09.06.2019 17:41:59     New time added: 7:00 - 9:00
```

Рисунок 5.9 – Запис усіх подій до файлу

ВИСНОВКИ

У ході аналізу існуючого програмного забезпечення моніторингу та управління енергетичними потоками було досліджено системи, які слугують для вирішення поставлених задач. Аналіз показав, що існуючі системи вирішують задачу не у повному обсязі, громіздкими та мають високі апаратні вимоги до пристроїв користувачів.

Розроблений програмний продукт є актуальним, оскільки розробляється як частина більшої системи, яка працює без участі людини. Розроблена система дозволяє легко інтегрувати додаткове обладнання, та його налаштувати.

Розроблений програмний продукт дозволяє автоматизувати роботу ланки електромережі та допомагає сповіщувати про неполадки чи збої системи.

Проведено огляд методів і засобів розробки програмної системи. Обґрунтовано вибір створення програмної системи, заснованої на веб-технологіях, а також побудованої за триланковою архітектурою. Це дає змогу підвищити гнучкість та зручність системи, як у розробці та супроводі, так і у використані.

За результатами виконання тестових завдань підтверджена коректність отриманих результатів, отже система відповідає поставленим вимогам.

Користувачами системи можуть бути різноманітні користувачі, які здійснюють нагляд за системою енергопостачання з метою виявлення надзвичайних ситуацій, які виникають, а також з метою швидкої реакції на дані ситуації.

Робота над дипломним проектом покращила знання різноманітних технологій, що використовуються під час розробки програмного забезпечення. Також було досліджено різноманітні техніки та алгоритми машинного навчання, було виявлено різноманітні сфери, для яких можна застосовувати дані прийоми. Також було створено декілька прототипів програмного забезпечення, які вирішували різноманітні аспекти поставленої задачі, а також які лягли в основу розробленого програмного забезпечення.

Програмне забезпечення дозволяє отримувати дані та відправляти команди обладнанню. Сконструйовано апаратно-програмний агент на базі Orange Pi та контролеру сімейства STM32.

Програмна система має клаєнт-серверну архітектуру. Для реалізаціїї сервісного шару системи миттєвого обміну повідомленнями, використане різноманітне програмне забезпечення: STM32CubeMx, System Workbench for STM32, Visual Studio 2017 та інші.

Створений програмний продукт дозволяє інтегрувати функції даного агенту в мультиагентну систему та працювати з іншими продуктами в одному середовищі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Liu, Xuan & Su, Bin. (2009). Microgrids - An integration of renewable energy technologies. 1 - 7. 10.1109/CICED.2008.5211651.

2. Foo, Yi Shyh Eddy & B. Gooi, H & Shuaixun, Chen. (2014). Multi-Agent System for Distributed Management of Microgrids. IEEE Transactions on Power Systems. PP. 1-11. 10.1109/TPWRS.2014.2322622.

3. Kubera, Yoann; Mathieu, Philippe; Picault, Sébastien (2010), «Everything can be Agent!» (PDF), Proceedings of the Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2010): 1547–1548

4. Проект Volttron [Електронний ресурс]. – Режим доступу: https://energy.gov/eere/buildings/volttron

5. Троелсен Э. Язык программирования C# 2010 и платформа .Net 4 / Э. Троелсен. –М. : Вильямс, 2011. – 1392 с.

6. RM0008 Reference manual [Електронний ресурс]. – Режим доступу: https://www.st.com/content/ccc/resource/technical/document/reference_manual/59/b9/ba/7f/11/af/43/d5/CD00171190.pdf/files/CD00171190.pdf/jcr:content/translations/en.CD00171190.pdf

7. Orange Pi PC User Manual [Електронний ресурс]. – Режим доступу: http://geekmatic.in.ua/pdf/OrangePi_PC_user_manual_v0.9.1.pdf

8. Arduino master. Orange Pi vs Raspberry Pi [Електронний ресурс]. – Режим доступу: https://arduinomaster.ru/raspberry-pi/orange-pi-konkurent-ili-podrazhatel-dlya-raspberry-pi/

9. Stroustrup, B. (2018). The C++ programming language. / B. – Addison-Wesley, - 2003. – 1366с.

10. Orange Pi. Wiring Op library. [Електронний ресурс]. – Режим доступу: http://www.orangepi.org/Docs/WiringPi.html

11. Генри С. Уоррен — Алгоритмические трюки для программистов / Генри С. Уоррен. — М.: Вильямс, 2014. — 512 с.

12. Блох Д. — Java. Эффективное программирование / Джошуа Блох — М. Лори, 2014. — 310 с.

13. STM32CubeMX [Електронний ресурс]. – Режим доступу: https://www.st.com/en/development-tools/stm32cubemx.html

ДОДАТОК 1

Розробка програмного агенту моніторингу та управління енергетичними потоками будівлі

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ1151188_18Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ1151188_18Б 81-1	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ1151188_18Б 12-1		Текс програмного продукту
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ1151188_18Б 12-2		Опис програмного модулю

ДОДАТОК 2

Розробка програмного агенту моніторингу та управління енергетичними потоками будівлі

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ1151188_18Б

Аркушів 13

Київ 2019

- 2 -

Основні функції прошивки контролеру STM32F103C8T6:

```
void On_light(int dec) {
	if (PWM_On_flag == 0) {
		for(int i=TIM4->CCR1; i<= dec; i+=1)
		{
			TIM4->CCR1=i;

			for(int d = 0; d < 100; d++) {}
		}
		PWM_On_flag = 1;
	} else
		TIM4->CCR1 = dec;
}

void Off_light(int dec) {
	if (PWM_On_flag == 1) {
		for(int i = TIM4->CCR1; i < 131072 - dec ;i+=1)
		{
			TIM4->CCR1= 131071-i;
			for (int d = 0; d < 100; d++) {}
		}
		PWM_On_flag = 0;
	}
}

void pwm(int32_t d) {
	if (TIM4->CCR1 < d)
		On_light(d);
	else
		Off_light(d);
}

void Led_on() {
	PWM_On_flag = 1;
}

void Led_off() {
	PWM_On_flag = 0;
}

static int	count_nums() {
	int i = 0;
	int num;

	num = counter;
	while ( num != 0) {
		i++;
		num /= 10;
	}
	return (i);
}

void increase_counter() {
	counter += 1;
}

char *zero_counter() {
	char	*buff;
	int		size;

	size = count_nums();
	buff = (char *)malloc(sizeof(char) * (size + 1));
	itoa(counter, buff, 10);

	buff[size] = '\0';

	return (buff);
}

void	send_message() {

	char	message[14];
	char	*num;
	int		size;
```

```
	message[0] = HAL_GPIO_ReadPin(A4_GPIO_Port, A4_Pin) == 0 ? '1' : '0';
	message[1] = ';';
	message[2] = HAL_GPIO_ReadPin(A5_GPIO_Port, A5_Pin) == 0 ? '1' : '0';
	message[3] = ';';
	message[4] = HAL_GPIO_ReadPin(A6_GPIO_Port, A6_Pin) == 0 ? '1' : '0';
	message[5] = ';';
	message[6] = HAL_GPIO_ReadPin(A7_GPIO_Port, A7_Pin) == 0 ? '1' : '0';
	message[7] = ';';

	num = zero_counter();
	size = strlen(num);

	if (size < 5) {
		size = 5 - size;
		for (int i = 8; i < 8 + size; i++) {
			message[i] = '0';
		}
	}

	for (int i = 8 + size, j = 0; i < 13; i++, j++) {
		message[i] = num[j];
	}

	message[13] = '\0';
	NVIC_DisableIRQ(EXTI1_IRQn);
	HAL_UART_Transmit(&huart1, (uint8_t *)message, 14, HAL_MAX_DELAY);
	counter = 0;
	NVIC_EnableIRQ (EXTI1_IRQn);

	bzero(message, strlen(message));
	free(num);

//	HAL_UART_Transmit(&huart1, (uint8_t *)message, 5, HAL_MAX_DELAY);

//	free(buff);
}

void	get_message() {
	char buffrec[BUFF_SIZE];

	bzero(buffrec, BUFF_SIZE);
	HAL_UART_Receive(&huart1, (uint8_t *)buffrec, 8, HAL_MAX_DELAY);

	if (strlen(buffrec) == 8)
	{
		if (buffrec[0] == '1')
			HAL_GPIO_WritePin(GPIOD, Com_Pin, GPIO_PIN_SET);
		else if (buffrec[0] == '0')
			HAL_GPIO_WritePin(GPIOD, Com_Pin, GPIO_PIN_RESET);
		else {}

		pwm(atoi(buffrec + 2));
	}
}

void EXTI_Init() {
	//тактирование AFIO & APB2ENR
	RCC->APB2ENR |= RCC_APB2ENR_IOPAEN;
	RCC->APB2ENR |= RCC_APB2ENR_AFIOEN;

	//сброс на 0
	GPIOA->CRH &= ~(GPIO_CRL_MODE1 | GPIO_CRL_CNF1);

	//CNF: режим Pull Up/Pull Down
	GPIOA->CRL |= (0x02 << GPIO_CRL_CNF0_Pos);
	//подтяжка вверх
	GPIOA->ODR |= (1 << 0);

	AFIO->EXTICR[1] &= ~(AFIO_EXTICR1_EXTI1);

	//прерывание по спаданию импульса
	EXTI->FTSR |= EXTI_FTSR_TR1;

	EXTI->PR = EXTI_PR_PR1;	//Сбрасываем флаг прерывания
```

```
                        //перед включением самого прерывания
        EXTI->IMR |= EXTI_IMR_MR1;   //Включаем прерывание 0-го канала EXTI

        NVIC_EnableIRQ(EXTI1_IRQn);
}
void EXTI1_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI1_IRQn 0 */
        NVIC_DisableIRQ(EXTI1_IRQn);
         if(__HAL_GPIO_EXTI_GET_IT(A1_Pin) != RESET)
          {
                EXTI->PR |= EXTI_PR_PR1;
                if ( HAL_GetTick() - old_time >= 100) {
                        increase_counter();
                        //HAL_GPIO_WritePin(Ld2_GPIO_Port, Ld2_Pin, GPIO_PIN_SET);
                        old_time = HAL_GetTick();
//              __HAL_GPIO_EXTI_CLEAR_IT(A1_Pin);

                }

//          HAL_GPIO_EXTI_Callback(A1_Pin);
         }

  /* USER CODE END EXTI1_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_1);
  /* USER CODE BEGIN EXTI1_IRQn 1 */
        NVIC_EnableIRQ (EXTI1_IRQn);
  /* USER CODE END EXTI1_IRQn 1 */
}
```

Основні функції серверу:

```
#include <iostream>
#include <cstdlib>
#include <cstring>
#include "PracticalSocket.h"
#include <pthread.h>
#include <stdint.h>
#include <semaphore.h>

#include "usart.h"

#define OK ok
#define LOG_FILE_PATH "log.txt"

using namespace std;

const int RCVBUFSIZE = 14;

char ok[]="acceptance";
void HandleTCPClient(TCPSocket *sock);
void *ThreadMain(void *arg);
void sendMessage(string IP, uint16_t port,char*  msg);
uint8_t equals(char first[],char second[]);
void *iostreamHandler(void* arg);
void threadSafeOutput(string msg);
void threadSafeConsoleOutput(string msg);
void *UsartMain(void *arg);

uint16_t ServerPort = 55556;
sem_t semaphore;
FILE *file;
string ip = "192.168.5.52";
uint16_t port = 55555;
char *mess;
char *uart_received;
char *tcp_received;
int fd;

int main(int argc, char **argv) {

 pthread_t iostreamThreadID;
```

```
sem_init(&semaphore, 0, 1);
//threadSafeConsoleOutput("Enter the server port:");
//cin >> ServerPort;
 wiringPiSetup();
 uint8_t *arg;
 arg=new uint8_t;
 pthread_t threadID_Usart;

*arg=0;
    if (pthread_create(&threadID_Usart, NULL, UsartMain, (void*)arg) != 0) {
                                cerr << "Unable to create thread" << endl;
        exit(1);
                }

  try{
    TCPServerSocket servSock(ServerPort);
    for (;;) {
      TCPSocket *clntSock = servSock.accept();
                threadSafeOutput("Socket accepted");
      pthread_t threadID;
      if (pthread_create(&threadID, NULL, ThreadMain,
            (void *) clntSock) != 0) {
        cerr << "Unable to create thread" << endl;
        exit(1);
      }

    }
  } catch (SocketException &e) {
    cerr << e.what() << endl;
    exit(1);
  }

}

void* UsartMain(void * arg) {

                int size;
//115200
    if ((fd = serialOpen ("/dev/ttyS3", 9600)) < 0)
    {
      fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
    }
                for (;;) {

if (serialDataAvail(fd) >= 14) {

//              threadSafeConsoleOutput("R:");
    uart_received = get_message(fd);
//              threadSafeConsoleOutput(uart_received);
                sendMessage(ip, port, uart_received);
                delete [] uart_received;
  }
}
}

void sendMessage(string IP, uint16_t port,char* msg){
 //threadSafeOutput("Creating TCP socket\n");
 TCPSocket *tempClient = new TCPSocket(IP,port);
 //threadSafeOutput("Sending message\n");
 tempClient->send(msg,strlen(msg));
 //threadSafeOutput("After send message\n");
  char echoBuffer[RCVBUFSIZE];
  int recvMsgSize = tempClient->recv(echoBuffer, RCVBUFSIZE);
  //threadSafeOutput(echoBuffer);
//threadSafeOutput("\n");
  if (equals(echoBuffer,OK)){
//threadSafeOutput("received acceptance");
 }
 else{
//threadSafeOutput("No ACK");
 }
//threadSafeOutput("continuing workflow");
delete tempClient;
//threadSafeOutput("closed TCPSocket");
```

```
}

void HandleTCPClient(TCPSocket *sock) {
threadSafeOutput("Handling client ");

  try {
      sock->getForeignAddress();
threadSafeOutput(sock->getForeignAddress());
threadSafeOutput(":");
  } catch (SocketException &e) {
    cerr << "Unable to get foreign address" << endl;
  }
  try {
      sock->getForeignPort();
threadSafeOutput(to_string(sock->getForeignPort()));
  } catch (SocketException &e) {
    cerr << "Unable to get foreign port" << endl;
  }
threadSafeOutput(" with thread ");
threadSafeOutput(to_string(pthread_self()));
threadSafeOutput("\n");
  char echoBuffer[RCVBUFSIZE];
  int recvMsgSize = sock->recv(echoBuffer, RCVBUFSIZE);
threadSafeOutput(echoBuffer);
threadSafeOutput("\n");
char* pok;
pok=ok;
  threadSafeOutput("Accepted");
  sock->send(pok,strlen(pok)+1);
 // send_message(fd, echoBuffer);
//sendMessage(fIP,fServerPort,"after receiving a message vlad replied with this\n");
//cout<<echoBuffer;
}

void *ThreadMain(void *clntSock) {
  pthread_detach(pthread_self());
threadSafeOutput("Before calling HandleTCPClient in ThreadMain\n");
  HandleTCPClient((TCPSocket *) clntSock);
  delete (TCPSocket *) clntSock;
  return NULL;
}


void *iostreamHandler(void* arg) {
//  pthread_detach(pthread_self());
//  uint16_t PORT;
//  while(1){
//  char msg[1024] ;
//  string  IP;
//threadSafeConsoleOutput("any message to send?\n");
//  cin>>msg;
//threadSafeConsoleOutput(msg);
//threadSafeConsoleOutput("\nIP:");
//  cin>>IP;
//threadSafeConsoleOutput(IP);
//threadSafeConsoleOutput("\nPORT:");
//  cin>>PORT;
//threadSafeConsoleOutput(to_string(PORT));
//threadSafeConsoleOutput("\n");
//  sendMessage(IP,PORT,msg);

//}
  return NULL;
}

void threadSafeOutput(string msg){
                sem_wait(&semaphore);
                file = fopen(LOG_FILE_PATH,"a");
                if(file == NULL){
        cerr << "Unable to open log file" << endl;
                exit(1);
}
                fputs(msg.c_str(), file);
                fputs("\n",file);
                fclose(file);
```

```
		sem_post(&semaphore);
}

void threadSafeConsoleOutput(string msg){
		sem_wait(&semaphore);
		cout << msg;
		sem_post(&semaphore);
}

uint8_t equals(char first[],char second []){
if (strlen(first)!=strlen(second))
    return 0;
  for(uint16_t i=0;i<strlen(first);i++){
    if (first[i]!=second[i]){
      return 0;
    }
  }
  return 1;
}
/*
boot
creating server socket
creating thread for console interface
*/

#include "usart.h"

void  send_message(int fd, char *str) {
		//size_t size;

		//size = strlen(str);
		for (size_t i = 0; i < 9;i++) {
	serialPutchar(fd, str[i]);
		}
}

char *get_message(int fd) {
		int size;
		int c;
		char *str;

		size = serialDataAvail(fd);
		if (size > 0) {
		    str = new char[size + 1];
	bzero(str, size);
				for(int i = 0; i < size; i++) {
					c = serialGetchar(fd);
					if (c <= 0 && c >= 255) {
						serialFlush(fd);
						break;
					}
					else if (c != -1)
					   str[i] = c;
					else {
					  serialFlush(fd);
					  break;
				    }
				}
				str[size] = '\0';
		}
		return(str);
}

/*int main() {
 int fd ;
// char *str;

  wiringPiSetup();
  if ((fd = serialOpen ("/dev/ttyS3", 115200)) < 0)
  {
    fprintf (stderr, "Unable to open serial device: %s\n", strerror (errno)) ;
    return 1 ;
  }
```

```
// Loop, getting and printing characters

 for (;;)
{
                printf("Sent1\n");
                serialPutchar(fd, '1');
   //send_message(fd, "1");
   delay(3000);
 //    str = get_message(fd);
/*     if (strstr(str, "EA")
       //send tcp/ip
     else
     {
                              //send packet
                }
     printf("%s", str);*/
/*
     printf("Sent0\n");
     serialPutchar(fd, '0');
//send_message(fd, "0");
   delay(3000);
  }
}*/
```

Код інтерфейс-додатку:

```
public partial class Form1 : Form
   {
      public string to_send_data;

      TcpListener listener;
      TcpClient receiver;

      ErrorHandler error = new ErrorHandler();

      string remoteAddress = "192.168.10.21";
      string LocalIP;

      int  pwm_set;
      bool com_on = false;

      int remotePort = 55556;
      int LocalPort = 55555;
      Random rand = new Random();

      double ticks;

      public Form1()
      {
         InitializeComponent();

         textBox1.TextAlign = HorizontalAlignment.Right;
         textBox2.TextAlign = HorizontalAlignment.Center;

         IPAddress[] localIP = Dns.GetHostAddresses(Dns.GetHostName());
         foreach (IPAddress address in localIP)
         {
            if (address.AddressFamily == AddressFamily.InterNetwork)
            {
               LocalIP = address.ToString();
            }
         }

         timer1.Start();
         timer1.Enabled = true;

         listener = new TcpListener(IPAddress.Parse(LocalIP), LocalPort);

         receiver = new TcpClient();

         Thread ServerMainThread = new Thread(ServerMain);
         ServerMainThread.Start();
```

```
    Thread ChartThread = new Thread(ChartUpdate);
    ChartThread.Start();

}

private void ChartUpdate(object chartUpdate)
{
    double[] chart = (double[])chartUpdate;
    DateTime minValue = DateTime.Now;
    DateTime maxValue = minValue.AddSeconds(120);

    chart1.ChartAreas[0].AxisX.Minimum = minValue.ToOADate();
    chart1.ChartAreas[0].AxisX.Maximum = maxValue.ToOADate();

    chart1.ChartAreas[0].AxisY.Maximum = 50;
    chart1.ChartAreas[0].AxisY.Minimum = 0;

    chart1.ChartAreas["ChartArea1"].AxisX.LabelStyle.Format = "HH:mm:ss";

    chart1.Series.Clear();

    Series newSeries = new Series("Series1");
    newSeries.ChartType = SeriesChartType.Line;
    newSeries.BorderWidth = 2;
    newSeries.Color = Color.Blue;
    newSeries.XValueType = ChartValueType.Time;
    chart1.Series.Add(newSeries);

    while (true)
    {

        if (chart1.IsHandleCreated)
        {
            this.Invoke((MethodInvoker)delegate { UpdateChart(); });
        }
        else
        {
            //.....
        }

        Thread.Sleep(1000);
    }
}

private void UpdateChart()
{
    DateTime timeStamp = DateTime.Now;

    foreach (Series ptSeries in chart1.Series)
    {
        AddNewPoint(timeStamp, ptSeries);
    }
}

  public void AddNewPoint(DateTime timeStamp, System.Windows.Forms.DataVisualization.Charting.Series ptSeries)
{
    double newVal = 0;

    ticks = 50 -  rand.Next(20, 30);

    if (ptSeries.Points.Count > 0)
    {
        newVal = ptSeries.Points[ptSeries.Points.Count - 1].YValues[0] + ticks;
    }

    if (newVal < 0)
        newVal = 0;
    ptSeries.Points.AddXY(timeStamp.ToOADate(), ticks);

    double removeBefore = timeStamp.AddSeconds((double)(90) * (-1)).ToOADate();
    while (ptSeries.Points[0].XValue < removeBefore)
    {
        ptSeries.Points.RemoveAt(0);
    }
```

```
    chart1.ChartAreas[0].AxisX.Minimum = ptSeries.Points[0].XValue;
    chart1.ChartAreas[0].AxisX.Maximum = DateTime.FromOADate(ptSeries.Points[0].XValue).AddMinutes(2).ToOADate();

    chart1.Invalidate();
}

private void ReceiveMessage(object rec)
{
    TcpClient receiver = (TcpClient)rec;
    char[] data = new char[255];
    string received_data = "";

    StreamReader STR = new StreamReader(receiver.GetStream(), System.Text.Encoding.ASCII);
    StreamWriter STW = new StreamWriter(receiver.GetStream(), System.Text.Encoding.ASCII);
    try
    {
        STR.Read(data, 0, data.Length);
        received_data = new string(data);
//      textBox3.Invoke((MethodInvoker)delegate { textBox3.Clear(); textBox3.AppendText(received_data);  });
        STW.Write("acceptance");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
    finally
    {
        parseData(received_data);
        STW.Close();
        STR.Close();
        receiver.Close();
    }
}

private void change_button(Button button, int on)
{
    if (on == 1)
    {
        button.FlatAppearance.BorderColor = System.Drawing.Color.Chartreuse;
        button.BackColor = System.Drawing.Color.Chartreuse;
        Invoke((MethodInvoker)delegate { to_console(button.Text + " is on ", null); });
    }
    else
    {
        button.FlatAppearance.BorderColor = System.Drawing.Color.Red;
        button.BackColor = System.Drawing.Color.Transparent;
        Invoke((MethodInvoker) delegate { to_console(button.Text + " is off ", null); });
    }
}

private void parseData(string received)
{
    string[] data = received.Split(';');

    consoleBox.Invoke((MethodInvoker)delegate { consoleBox.AppendText(received); consoleBox.AppendText(Environment.NewLine); });

    if (data.Length == 5)
    {

        if (string.Equals(data[0], "1"))
            change_button(button1, 1);
        else
            change_button(button1, 0);

        if (string.Equals(data[1], "1"))
            change_button(button2, 1);
        else
            change_button(button2, 0);

        if (string.Equals(data[2], "1"))
            change_button(button3, 1);
        else
            change_button(button3, 0);

        if (string.Equals(data[3], "1"))
```

```
            change_button(button4, 1);
        else
            change_button(button4, 0);

        if (data[4].Length > 0)
            ticks = double.Parse(data[4]);

    }

}

private void ServerMain()
{
    listener.Start();

    while (true)
    {
        try
        {
            receiver = listener.AcceptTcpClient();
            if (receiver.Connected)
            {
                Thread receiveThread = new Thread(ReceiveMessage);
                receiveThread.Start(receiver);
            }

        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.ToString());
        }
    }

}

private void to_console(string message, Exception ex)
{
    if (ex != null)
    {
        consoleBox.AppendText(message + ex.Message);
        consoleBox.AppendText(Environment.NewLine);
        error.write_exception(ex);
    }
    else
    {
        consoleBox.AppendText(message);
        consoleBox.AppendText(Environment.NewLine);
        error.write_error(message);
    }

}

private void button7_Click(object sender, EventArgs e)
{
    char [] message = new char[12];
    TcpClient send = new TcpClient();

    if (com_on == true)
    {
        to_send_data = "1";
        Invoke((MethodInvoker)delegate { to_console("Communication: On ", null); });
    }
    else
    {
        to_send_data = "0";
        Invoke((MethodInvoker)delegate { to_console("Communication: Off ", null); });
    }

    to_send_data += ";" + pwm_set.ToString() + "\0";

    Invoke((MethodInvoker)delegate { to_console("PWM set: " + pwm_set.ToString(), null); });

    try
    {
        send.Connect(remoteAddress, remotePort);
```

```
            }
            catch (Exception ex)
            {
                Invoke((MethodInvoker)delegate { to_console("Connection failed.", null); });
                Invoke((MethodInvoker)delegate { to_console("Error occured: ", ex); });
                return;
            }

            StreamWriter STW = new StreamWriter(send.GetStream(), System.Text.Encoding.ASCII);
            StreamReader STR = new StreamReader(send.GetStream(), System.Text.Encoding.ASCII);

            try
            {
                STW.Write(to_send_data);
                consoleBox.Invoke((MethodInvoker)delegate { consoleBox.AppendText(to_send_data);
consoleBox.AppendText(Environment.NewLine); });
                //STR.Read(message, 0, message.Length);
                //if (string.Equals(message.ToString(), "acceptance") != true)
                //{
                //    textBox3.Invoke((MethodInvoker)delegate { textBox3.AppendText(" and was " + message.ToString() + "d");
textBox3.AppendText(Environment.NewLine); });
                //}
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.ToString());
            }
            finally
            {
                STW.Close();
                STR.Close();
                send.Close();
            }

        }

        //set PWM
        private void button6_Click(object sender, EventArgs e)
        {

            pwm_set = trackBar1.Value;

        }

        private void Form1_FormClosing_1(object sender, FormClosingEventArgs e)
        {
            Application.Exit();
        }

        private void button5_Click_2(object sender, EventArgs e)
        {
            if (button5.Text == "On")
            {
                button5.Text = "Off";
                button5.BackColor = System.Drawing.Color.Red;
                button8.FlatAppearance.BorderColor = System.Drawing.Color.Chartreuse;
                button8.BackColor = System.Drawing.Color.Chartreuse;
                com_on = true;

            } else
            {
                button5.Text = "On";
                button5.BackColor = System.Drawing.Color.Chartreuse;
                button8.FlatAppearance.BorderColor = System.Drawing.Color.Red;
                button8.BackColor = System.Drawing.Color.Red;
                com_on = false;

            }

        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            double per = (double)trackBar1.Value  * 100.0 / (double)trackBar1.Maximum;
```

```
        textBox1.Text = per.ToString("0.000");
        textBox2.Text = per.ToString("0.000");
    }

    private void tabPage1_Click(object sender, EventArgs e)
    {

    }

    private void label10_Click(object sender, EventArgs e)
    {


    }

    private void timer1_Tick(object sender, EventArgs e)
    {
        if (button9.BackColor == System.Drawing.Color.Red)
        {
            button9.BackColor = System.Drawing.Color.White;
        }

        if (button9.BackColor == System.Drawing.Color.White)
        {
            button9.BackColor = System.Drawing.Color.Red;
        }
    }

    private void label1_Click(object sender, EventArgs e)
    {

    }

    private void radioButton2_CheckedChanged(object sender, EventArgs e)
    {
        groupBox1.Enabled = false;
        groupBox2.Enabled = true;
    }

    private void label5_Click(object sender, EventArgs e)
    {

    }

    private void radioButton1_CheckedChanged(object sender, EventArgs e)
    {


        groupBox1.Enabled = true;
        groupBox2.Enabled = false;
    }

    //Add
    private void button10_Click(object sender, EventArgs e)
    {
        try
        {
            DateTime fr = DateTime.Parse(textBox4.Text);
            DateTime to = DateTime.Parse(textBox5.Text);

            textBox3.AppendText(fr.ToShortTimeString() + " - " + to.ToShortTimeString() + ";");
            textBox3.AppendText(Environment.NewLine);
            Invoke((MethodInvoker) delegate { to_console("New time added: " + fr.ToShortTimeString() + " - " + to.ToShortTimeString(), null); });

        } catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

ДОДАТОК 3

Розробка програмного агенту моніторингу та управління енергетичними потоками будівлі

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТІ1151188_18Б

Аркушів

Київ 2019

АНОТАЦІЯ

Розроблений програмний продукт забезпечує керування та моніторинг енергетичних потоків будівлі.

Дипломна робота присвячена розробці програмного агента моніторингу та управління системи вентиляції будівлі на базі Orange Pi та STM32, зпсобами платформи Windows Forms, STM32CubeMx, System Wokrbench for STM, Visual Studio 2017 для подальшого використання у мультиагентній системі.

ЗМІСТ

1 ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

Клієнт-додаток називається PowerClient. Даний продукт розорблено у середовищі Visual Studio 2017, використовуючи об'єктно-орієнтовану мову програмування C#, фреймворк .NET. Прошивка мікроконтролеру була розроблена на мові C, застосунок на одноплатному комп’ютері – C++.

Програма призначена для моніторингу та управління енергетичними потоками будівлі.

1.1 Опис логічної структури

Програмний продукт було розроблено у вигляді декстопного додатку за допомогою інструментів Windows Forms.

Данна система складається з 3 основних частин: прошивка мікроконтролеру, консольний додаток на Orange Pi та клієнт-додаток.

Прошивка мікроконтролеру складається з окремих логічних блоків для збирання кожного типу даних. Також тримає окремі блоки отримання команд та відправлення інформації.

Консольний додаток отримує, аналізує інформацію та має оремі блоки для обміну інформацією з мікроконтролером та з клієнт-додатком.

У разі вдалої роботи програми, користувач буде здатний спостерігати стан показового стенду через клієнт-додаток та керувати його роботою.

1.2 Вхідні та вихідні дані

Вихідними даними для контролеру є дані з датчиків та значення, знаяті з лічильнику.

Вхідними даними для контролеру є команди включення/виключення установки, подання широтно-імпульсної модуляції.

Вхідними даними для серверного додатку є дані з контролеру та команди з клієнт-додатку.

Вихідними даними є команди для контролеру та поточні значення системи для клієнт-додатку.

Вхідними даними для клієнт додатку є дані з серверного додатку та команди від користувача.

Вихідними даними для клієнт додатку є команди, введені користувачем.

ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний модуль було про тестовано на комп’ютері, який побудовано на базі процесору архітектури x64 Intel Core i5 та має 16 Гб оперативної пам’яті. Розроблене програмне забезпечення не потребує значних апаратних витрат, тому може використовуватись на менш потужних комп’ютерах, оскількт всі дії відбуваються на серверному додатку.